

AD-A118 733

PAR TECHNOLOGY CORP NEW HARTFORD NY
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. USE--ETC(U)
JUN 82 S E HAENN, D MORRIS

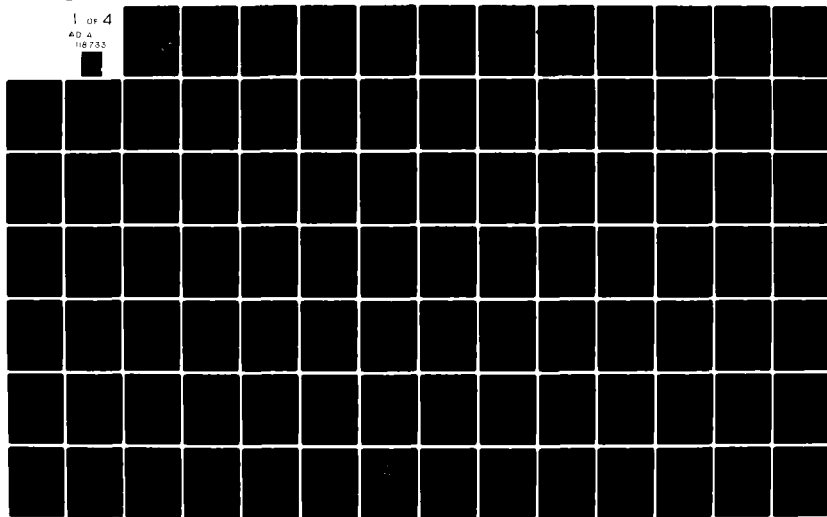
F/G 9/2

UNCLASSIFIED PAR-62-21

NL

1 of 4

AD A
118733



2



PAR TECHNOLOGY CORPORATION

AD A118733

DTIC FILE COPY

DTIC
ELECTE
AUG 31 1982
H

Approved for public release:
distribution unlimited

82 08 20 02 8

OLFARS User Manual

June 18, 1982

DTIC
COLLECTED
AUG 31 1982
H

DTIC
COPY
INSPECTED
2

| | |
|--------------------|-------------------------------------|
| Accession For | |
| NTIS GRA&I | <input checked="" type="checkbox"/> |
| DTIC TAB | |
| Unannounced | |
| Justification | |
| By | |
| Distribution/ | |
| Availability Codes | |
| Avail and/or | |
| Dist | Special |
| A | |

OLPARS User Manual

Submitted By

PAR TECHNOLOGY CORPORATION
Route 5, Seneca Plaza
New Hartford, New York 13413

Authors

Mr. Steven E. Haehn
Ms. Donna A. Morris

PAR REPORT #82-21

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|--|-----------------------|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| | AD-A118723 | |
| 4. TITLE (and Subtitle) | | 5. TYPE OF REPORT & PERIOD COVERED |
| OLPARS VI (On-Line Pattern Analysis and Recognition System) | | USER'S MANUAL |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| | | 82-21 |
| 7. AUTHOR(s) | | 8. CONTRACT OR GRANT NUMBER(s) |
| Mr. Steven E. Haehn Ms. Donna Morris | | |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| PAR Technology Corporation Route #5, Seneca Plaza New Hartford, New York 13414 | | |
| 11. CONTROLLING OFFICE NAME AND ADDRESS | | 12. REPORT DATE |
| Department of Defense Washington, D. C. | | June 18, 1982 |
| | | 13. NUMBER OF PAGES |
| | | |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) |
| Same As Block 11 | | Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |
| | | |
| 16. DISTRIBUTION STATEMENT (of this Report) | | |
| Approved for public release; distribution unlimited | | |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) | | |
| | | |
| 18. SUPPLEMENTARY NOTES | | |
| | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) | | |
| Pattern Recognition, Structure Analysis, Discriminant Analysis, Data Transformation, Feature Extraction, Feature Evaluation, Cluster Analysis, Classification Computer Software | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) | | |
| <p>The User's Manual introduces the reader to OLPARS concepts, some pattern recognition methodology, and the OLPARS displays. Instructions example usage of each OLPARS command is included.</p> | | |

TABLE OF CONTENTS

| | | |
|---------|---|----|
| 1. | ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM . . . | 1 |
| 1.0 | INTRODUCTION | 1 |
| 2. | FUNCTIONAL ASPECTS OF OLPARS | 3 |
| 2.0 | INTRODUCTION | 3 |
| 2.1 | OLPARS DATA REPRESENTATION | 3 |
| 2.1.1 | Trees, Nodes, And "What Is Current" | 4 |
| 2.1.2 | OLPARS Current Option | 6 |
| 2.1.3 | Excess Measurement Mode | 6 |
| 2.1.4 | Ignored Measurements | 8 |
| 2.1.5 | Vector Identifiers | 8 |
| 2.1.6 | Notes On User Interaction With Commands | 9 |
| 2.2 | STATISTICAL VALIDITY OF USER DATA SET | 10 |
| 2.3 | DISPLAY DESCRIPTIONS | 11 |
| 2.3.1 | One-Space Displays (MICRO And MACRO) | 14 |
| 2.3.1.1 | One-Space Display Format | 14 |
| 2.3.2 | Two-Space Displays (SCATTER And CLUSTER) | 19 |
| 2.3.2.1 | Two-Space Display Format | 19 |
| 2.3.3 | Confusion Matrix Displays | 24 |
| 2.3.4 | Rank Order Displays | 32 |
| 2.3.4.1 | Ranking Information | 35 |
| 2.3.5 | Data Tree Structural Displays | 36 |
| 2.3.6 | Logic Tree Structural Displays | 40 |
| 2.3.6.1 | Logic Node Information | 42 |

TABLE OF CONTENTS (continued)

| | | |
|---------|--|-----|
| 3. | USING OLPARS | 47 |
| 3.0 | YOUR'RE NEW, SO WHAT DO YOU DO? | 47 |
| 3.1 | NOTES ON DATA COLLECTION | 47 |
| 3.2 | IMPORTANCE OF FEATURE EXTRACTION | 48 |
| 3.3 | HELLO OLPARS | 49 |
| 3.4 | CREATING A DATA TREE | 50 |
| 3.5 | BEGINNING ANALYSIS - EXAMINING YOUR DATA | 50 |
| 3.6 | TEST DATA SET A "MUST" | 51 |
| 3.7 | STUDYING THE DATA STRUCTURE | 53 |
| 3.8 | RESTRUCTURING A DATA TREE, WHY? | 57 |
| 3.9 | CHOOSING THE "BEST" MEASUREMENTS | 61 |
| 3.9.1 | Selection Of Measurement By The Analyst | 65 |
| 3.9.1.1 | "Automatic" Selection Of Measurements | 66 |
| 3.9.1.2 | Measurement Reduction Transformation | 70 |
| 3.9.1.3 | A Final Note About Measurement Evaluation | 75 |
| 3.9.2 | Data Transformations | 76 |
| 3.9.2.1 | The Normalization Transformation | 76 |
| 3.9.2.2 | Eigenvector Transformation (Data Reduction) | 76 |
| 3.9.2.3 | Measurement Transformation | 80 |
| 3.10 | LOGIC DESIGN AND EVALUATION | 81 |
| 3.10.1 | Designing Logic | 82 |
| 3.10.2 | "Between-Group Logic" | 85 |
| 3.10.3 | "Within-Group" Logic | 90 |
| 3.10.4 | Testing The Logic (Overall Logic Evaluation) | 94 |
| 3.10.5 | Reassociated Names | 97 |
| 3.10.6 | Accepting Or Rejecting A Logic Design | 100 |

TABLE OF CONTENTS (continued)

| | | |
|--------|---|-----|
| 3.10.7 | Viewing Your Logic | 103 |
| 3.11 | SUMMARY | 103 |
| 4. | OLPARS COMMANDS | 107 |
| 4.0 | GENERAL STRUCTURE | 107 |
| 4.1 | COMMAND SUMMARY | 112 |
| 4.2 | COMMAND DESCRIPTIONS | 116 |
| 4.3 | OLPARS COMMAND DESCRIPTIONS. | 118 |
| | REFERENCES | 314 |
| | GLOSSARY | 315 |
| | INDEX. | I-1 |
| | APPENDIX A - Multics OLPARS Mathematics | A-1 |
| | APPENDIX B - OLPARS User Notes | B-1 |
| B.1 | Data Partitions (Boundaries) | B-1 |
| B.2 | Evaluation Procedure | B-2 |
| B.2.1 | One-Space. | B-2 |
| B.2.2 | Two-Space. | B-4 |
| B.2.3 | Examples | B-6 |

List Of Figures

| Figure | | Page |
|--------|--|------|
| 2-1 | OLPARS Command and Display Relationships. | 13 |
| 2-2 | General Layout Of One-Space Display. | 15 |
| 2-3 | One-Space Macro and Micro Views | 18 |
| 2-4 | General Format of Two-Space Display. | 21 |
| 2-5 | Two-Space Scatter and Cluster Plots. | 23 |
| 2-6 | General Format Of Between-Group Confusion Matrix. | 26 |
| 2-7 | Between-Group Confusion Matrix Example. | 27 |
| 2-8 | General Format Of Within-Group Confusion Matrix. | 30 |
| 2-9 | Within-Group Confusion Matrix Example. | 31 |
| 2-10 | General Format of Rank Order Display. | 33 |
| 2-11 | General Format Of Data Tree Display. | 38 |
| 2-12 | Data Tree Display Example. | 39 |
| 2-13 | General Format Of Logic Tree Display. | 41 |
| 2-14 | General Format Of Logic Tree Nodes and Logic Node Example. | 44 |
| 2-15 | Logic Tree Display Example | 45 |
| 3-1 | Some Available Information From PRTDS Command | 52 |
| 3-2 | One-Space Coordinate Projections | 55 |

List Of Figures

| | | |
|------|--|----|
| 3-3 | Eigenvalues And Two-space Eigenvector Projection. . . . | 56 |
| 3-4 | Two-Space Eigenvector Pro- jection Scatter Plots (With "selected" classes and Pro- jected class means) | 58 |
| 3-5 | Data Partitionment On Two-Space Displays | 59 |
| 3-6 | Restructured Data Tree. . . . | 60 |
| 3-7 | Two-Space Data Projection After Restructuring. | 63 |
| 3-8 | Overall Rank Order Display Restructuring (top showing vector count weighting) (bottom showing equalized weighting). | 64 |
| 3-9 | Rank Order Display For Single Class | 67 |
| 3-10 | Scatter Plots Showing Class Overlap. | 68 |
| 3-11 | Rank Order Displays For Class Pairs | 69 |
| 3-12 | Rank Order Displays Showing Measurement Selection | 71 |
| 3-13 | Rank Order Display Of Measurements To Be Used In A Data Transformation. | 73 |
| 3-14 | Data Tree Display And Scatter Plot Projection After Measurement Reduction | 74 |
| 3-15 | Eigenvalues And Data Tree After An Eigenvector Transformation . | 78 |
| 3-16 | Coordinate Projection Plot Of A Data Tree After An Eigenvector Transformation . . | 79 |
| 3-17 | Logic Design Choices | 83 |
| 3-18 | Logic Tree Formed Via NAMELOG . | 84 |

List Of Figures

| | | |
|-------|--|-------------|
| 3-19 | Creating Between-Group Logic On Two-Space Display | 86 |
| 3-20 | Logic Tree After Between-Group Logic Generation. | 88 |
| 3-21 | Creating Between-Group Logic On Two-Space Display | 89 |
| 3-22 | Within-Group Confusion Matrices | 92 |
| 3-23A | "Completed" Logic Tree. | 93 |
| 3-23B | (continued) | 94 |
| 3-24 | Test Set Data Trees. | 95 |
| 3-25 | Viewing Reassociated Class Names | 99 |
| 3-26 | An Overall Logic Evaluation | 101 |
| 4-1 | Command Categories In CLPARS. | 110 |
| 4-2 | CLPARS Command Categories. | 111 |
| 4-3 | CLPARS Command Description Format | 117 |
| B-1 | User Defined Regions On One And Two Space Displays. | B-3 |
| B-2 | Examples Of Vector-To-Region Assignments | B-7,B-8,B-9 |

SECTION 1

ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM (OLPARS)

1.0 INTRODUCTION*

A "pattern recognition" problem is described as the recognition of the state of an environment based on a set of features or measurements extracted from the environment. Therefore, a "pattern recognition" problem consists of (1) feature extraction, that is, the definition of the measurements used to describe an environment, and of (2) pattern classification.

Feature extraction should provide a set of measurements that yield information which aids in discriminating between the various environmental states. Pattern classification requires the design of a recognition logic which classifies the state of the environment using the extracted environmental features.

The concept of a vector space is fundamental to all of the problems discussed here. The extracted environmental features (from now on referred to as measurements) define the basis of the space; an object or an event is represented as a vector in that space, and pattern classification involves defining the partitionment of this space into regions associated with each of

* taken from [1] p.1-3

ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM -- 1
Introduction

the states (or classes) of the environment. In order to solve a pattern classification problem, statistical sample vectors from each class must be collected and analyzed to yield a satisfactory classification logic.

"Pattern analysis" differs from pattern classification in that the states (or classes) of the environment are not previously known to the researcher. The data consists of a set of vectors (measurements from the environment) which must be analyzed to determine the inherent classes contained within the vector data. The assumption made in pattern analysis is that vectors from a class will cluster together in the vector space. Detecting and identifying these clusters is the essence of pattern analysis.

SECTION 2

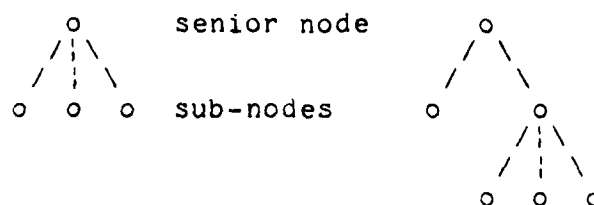
FUNCTIONAL ASPECTS OF OLPARS

2.0 INTRODUCTION

In the following sections, OLPARS data representation (trees, nodes) and operational concepts (current option) are described in detail. The descriptions of individual OLPARS displays can be found here. Included is a discussion of the statistical validity of a data set.

2.1 OLPARS DATA REPRESENTATION

The vector data structure is represented within OLPARS as a hierarchical tree where each node in the tree corresponds to a list of vectors. The data tree usually consists of a senior node (the root node of the tree, or oldest ancestor in the tree), with subnodes (leaves, or children) directly beneath the senior node.



FUNCTIONAL ASPECT OF OLPARS -- 2

OLPARS DATA REPRESENTATION

The senior node is associated with the entire list of vectors while the lower-order nodes represent sub-lists. In turn, each vector list associated with a child of a parent node may be sub-divided again (using structure analysis techniques), giving more nodes. Each subnode represents a single data class. (state in the environment).

The data vectors contain the measurements the user has collected outside of OLPARS. Each vector has an identifier (number) that can be user assigned or automatically generated (see FILEIN command). These identifiers give the user the ability to access individual vectors when the need arises.

2.1.1 Trees, Nodes, And "What Is Current" -

The names of data trees (and logic trees) are limited to a maximum of eight characters. The tree name must begin with an alphabetic character. The remaining letters (if any) must be alphanumeric. Data tree node names are limited to four characters. Any "printable"* character may be used in a data node name, except a comma (,). The first character of the data node name is referred to as the class display symbol, or class symbol. This symbol is used to identify vectors belonging to a given node or class. Logic nodes are referred to through numbers.

* does not include blanks, tabs, linefeeds etc.

FUNCTIONAL ASPECT OF OLPARS -- 2
Trees, Nodes, And "What Is Current"

| Valid data node names | Valid tree names |
|-----------------------|------------------|
| ----- | ----- |
| **** - Senior node | ABCDEFGH |
| ABCD | D |
| B | A1 |
| da! | GRAINS 2 |
| #ab | |

Four asterisks (****) are explicitly reserved to name the senior node of all data trees. The senior node of a logic tree is always named "1". Throughout OLPARS the concept of "current data set" is used. This refers to the data that the user has most recently designated for processing (see SETDS command). The name of the "current data set" is derived from a tree name - node name pair. (e.g. the "current data set" in the above table could be ABCDEFGH(****) or ABCDEFGH(ABCD), or ABCDEFGH(#ab)). When the "current data set" includes the senior node, the entire set of vectors in the tree are used. When a lower-order node is designated as the "current data set", only those vectors associated with that node are used. Whether the set contains the data under a single node of a preselected tree, or all of the data associated with the entire tree, the "current data set" is related to only one data tree.

Since more than one logic may exist for any single data set, the concept of "current logic" must be introduced. The "current logic" for a data set is the latest logic evaluated on the data set. Note that the "current logic" may exist for any data set, but

FUNCTIONAL ASPECT OF OLPARS -- 2

Trees, Nodes, And "What Is Current"

can only be created or evaluated on the "current data set".

2.1.2 OLPARS Current Option -

The OLPARS "current option" is used to determine which menu (option list) of suggested "next-commands" should be displayed to the user. Thus, the "current option" can be thought of as the "state" or "frame" in which OLPARS resides (e.g. structure analysis frame or logic design state).

User functions which alter the OLPARS current option consist of:

1. changing or deleting the current data set.
2. deleting the current logic tree
3. creating data projections
4. creating classification logic
5. performing measurement evaluations

The "current option" appears on all OLPARS displays. The commands which alter the "current option" are considered "major" OLPARS commands.

2.1.3 Excess Measurement Mode -

A data set is in "excess measurement mode" if the number of features contained in a vector is greater than fifty (the maximum number of measurements allowed is one hundred fifty). When such a data set is created in OLPARS (via FILEIN or MEASXFRM), the means

FUNCTIONAL ASPECT OF OLPARS -- 2
Excess Measurement Mode

and covariances are not computed for the OLPARS data tree.

Only a few programs are allowed to operate on data trees in "excess" measurement mode, they include:

S1CRDV Structure analysis using data coordinate projections

S2CRDV

L1CRDV Logic design using data coordinate projections

L2CRDV

FILEIN

FILEOUT Data Tree Utilities

MAKETREE

PRTDS

DSCRMEAS

MEASXFRM

SLCTMEAS Measurement evaluation and transformation commands

TRANSFRM

UNION

Once the number of features in an "excess measurement mode" data set has been reduced to fifty or less (via MEASXFRM or TRANSFRM), the means and covariance are calculated for that data set, and any OLPARS command can be used on the data set.

2.1.4 Ignored Measurements -

An analyst may find that particular measurements describing the environment being studied are not providing useful information for class separation. There might also be measurements that make analysis difficult.* Possibly, the analyst may want to see the class separation results without using some measurement(s). If any of these situations arise, it is possible to ignore some measurements of a data set, (during logic design or structure analysis) without physically reducing the number of measurements in the data set. (i.e., the measurements are eliminated from the logic design or structure analysis computations only; not the user data set). Further reference to ignored measurements can be found in the individual logic or structure analysis command descriptions.

2.1.5 Vector Identifiers -

Numeric values are assigned to each vector for purposes of identification (e.g. finding out which vector is not clustering properly in the structure analysis or misclassified during logic design). These numeric values (referred to as vector identifiers or vector ids.) can be pre-assigned to feature vectors before they enter OLPARS by the analyst, or as a data tree is created by OLPARS (see FILEIN command description).

When new data trees are created in OLPARS, the creating

* The covariance matrix statistics, calculated for every data class, may be a "singular matrix".

commands may ask if vector identifiers are to be resequenced. If no particular vector identifier has been assigned to a vector (by the analyst), then vector identifiers can be resequenced. Resequencing vector identifiers guarantees that a unique identifier is present for each vector in a class (i.e., vector identifiers are unique within a class, but may not be unique in the data tree). Note, there is no requirement that these vector identifiers be unique (i.e., OLPARS does not check for vector identifier uniqueness). However, this may lead to vector identification problems later.

2.1.6 Notes On User Interaction With Commands -

OLPARS commands can prompt the user for controlling information or parameters using either a long or short prompt. Commands will normally query a user with short prompts. The short prompt is a terse or abbreviated statement of the desired input. The user can change the prompt made from short prompts to long prompts, or vice versa, by invoking the command CDEFAULT.

As an additional user aid, the user can respond to any text prompt with a question mark (?). If the short prompt was initially displayed, the OLPARS command will respond by displaying the long prompt. If the long prompt was first displayed, the OLPARS command will display a message indicating that the user should try the HELP command for additional aid, and reprompt the user with the short prompt (Note, graphics prompts do not use this convention).

FUNCTIONAL ASPECT OF OLPARS -- 2

Notes On User Interaction With Commands

To escape an OLPARS command during user interaction, the user can enter a null response (carriage return) to a textual prompt, or a "q" (for quit) to a graphics prompt. Upon detecting this "quit" response the command will stop execution and return to the general command level of input (Note, there are a few exceptions to this case which will send the user to the next prompt. At the next prompt the user can "quit").

There are several OLPARS commands that make use of "class selection lists". The user views a list of class names that are under consideration and chooses which classes are to be used by the command via the class display symbol. If the user decides all classes are to be used, an asterisk (*) may be substituted for the list of class symbols. An initial minus sign (-) to the "class selection" prompt, before the class symbols, informs the command that every class NOT in the user supplied list is to be used by the command.

2.2 STATISTICAL VALIDITY OF USER DATA SET

(Or Will My Data Work In OLPARS)

As a user of OLPARS, the researcher must be aware of the statistical validity of the design data set being used to create classification logic, that is, how many vector samples are needed for an adequate classification design. If not enough samples are given in the design data set, the error rate of classification can be high using a test data set. It is even feasible that the logic

design algorithms will fail on the design data set (e.g. the inability to invert a covariance matrix).

D. Foley [3] shows that the estimate of the true error rate of a classifier is a function of the sample size per class (N) and feature size (L). In particular, the function used is the ratio of the sample size per class to the feature size (N/L). According to Foley, if the ratio of N/L is greater than three, then on the average the design-set error rate of classification is reasonably close to the test-set error rate. His results were obtained using a design data set with an underlying multivariate normal distribution, but the results may be more general. Foley also noted that if less is known about the underlying probability structure of the environment (data set), then the ratio of N/L should be increased.

Foley shows that the variance of the design-set error rate is approximated by a function that is bounded by $1/8N$. This implies that even if the number of features is small, enough samples must be used to minimize the variance of the design data set error rate.

2.3 DISPLAY DESCRIPTIONS

There are six major types of OLPARS displays: (1) one-space (micro or macro), (2) two-space (scatter or cluster), (3) confusion matrix (between group, within group), (4) rank order, (5) data tree, and (6) logic tree. The following sections describe each type of display. The descriptions are in the form of "display

FUNCTIONAL ASPECT OF OLPARS -- 2

DISPLAY DESCRIPTIONS

field entity - definition". There may be letters found in parenthesis, preceding the display field entity. These letters indicate the general region in which the entity can be found on the display ("L", as the first character, represents "lower"; as the second or only character, it represents "left"; "U" represents "upper"; "R" represents "right"; "C" represents "center"). Figure 2-1 gives a list of commands that either generates or modifies each type of display.

One and two space displays are created by structure analysis and group logic design commands. These commands present a "picture" of the structural or grouping nature of a data set.

Confusion matrix displays are created by logic design commands or the logic evaluation command. They summarize, within a table, the results of a partial or an overall logic evaluation, respectively.

Rank order displays are created by measurement evaluation routines. They present a ranking of measurement numbers, classes, or class pairs, and give an indication of the discriminatory power of the measurements in a data set.

Data and logic tree displays present a structural picture of data and logic trees, respectively.

| One-Space | Two-Space |
|------------------|------------|
| ----- | ----- |
| BINWIDTH | CDISPLAY |
| CDISPLAY | CSCALE |
| CSCALE | DBNDY |
| DBNDY | DRAWBNDY |
| DRAWBNDY | L2ASDG |
| INTENSIFY | L2CRDV |
| L1ASDG | L2EIGV |
| L1CRDV | L2FSHP |
| L1EIGV | PROJMN |
| PROJMN | RDISPLAY |
| RDISPLAY | REDRAW |
| REDRAW | REPROJECT |
| REPROJECT | RESTRUCT |
| RESTRUCT | S2CRDV |
| S1CRDV | S2EIGV |
| S1EIGV | S2FSHP |
| SCALRET | SCALRET |
| SCALZM | SCALZM |
| SELECT | SELECT |
| Confusion Matrix | Rank Order |
| ----- | ----- |
| CREATLOG | DSCRMEAS |
| LOGEVAL | RANK |
| NMEVAL | SLCTMEAS |
| PWEVAL | UNION |
| SUMMCM | |
| Data Tree | Logic Tree |
| ----- | ----- |
| DRAWTREE | DRAWLOG |

Figure 2-1 OLPARS Command And Display Relationships

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (one-space)

2.3.1 One-Space Displays (MICRO And MACRO) -

A one-space micro display gives a "combined" view of selected data class frequency histograms, that is, a frequency histogram from each projected data class is superimposed on a single base line. By using the INTENSIFY command, selected class histograms can appear as bargraphs.

A one-space macro display gives "individual" views of selected frequency histograms in a "stack histogram" format (each histogram has its own baseline). With more than three classes, a macro plot will look less cluttered than a micro plot.

2.3.1.1 One-Space Display Format -

A one-space display format contains the data-classes-present list, plot type, measurements used, date, time, menu, current data set name, current option name, and a user interaction notes area (see Figure 2-2). The data range (minimum and maximum data values), bin size, and bin count appear in the lower left region of the one-space display.

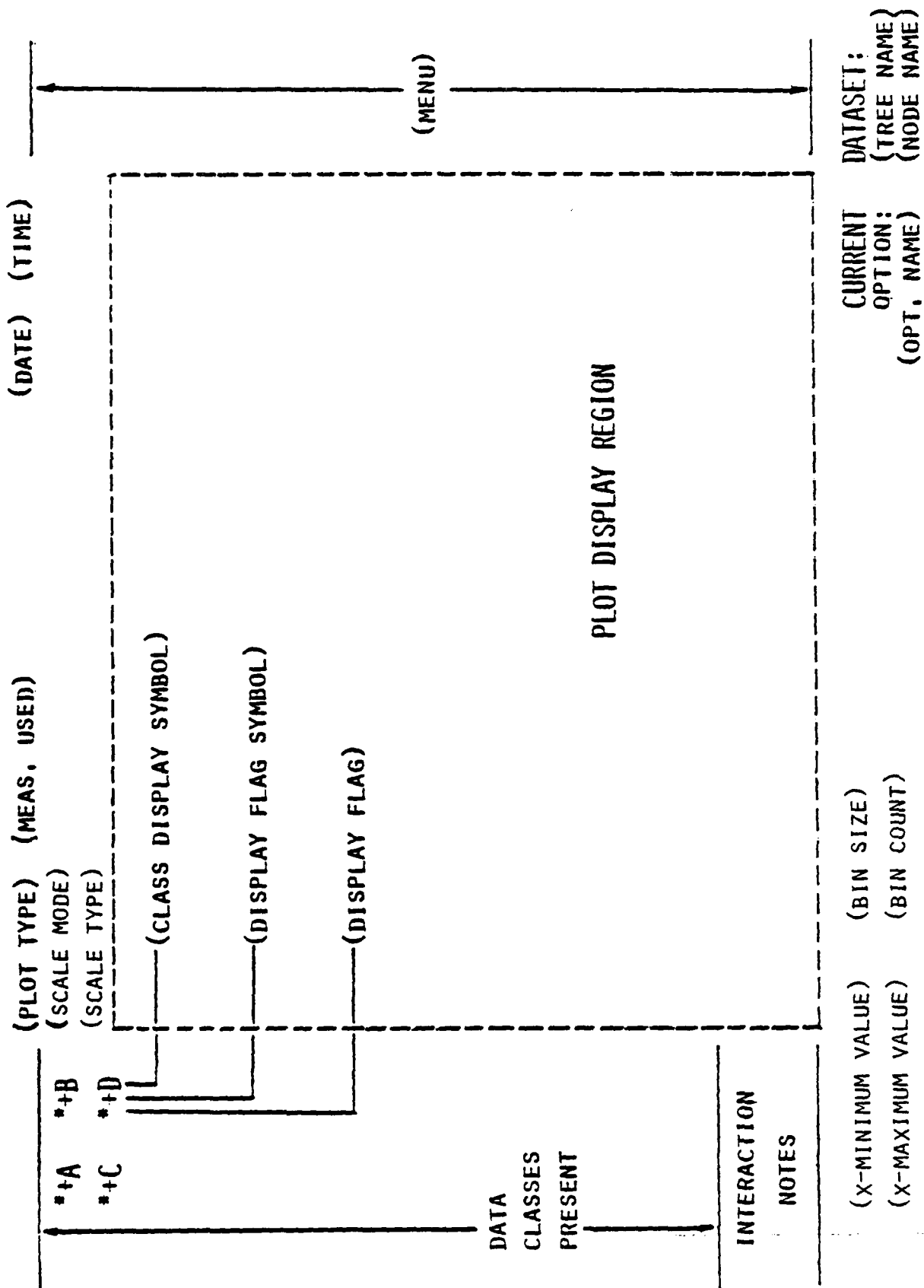


Figure 2-2 General Layout of One-Space Display

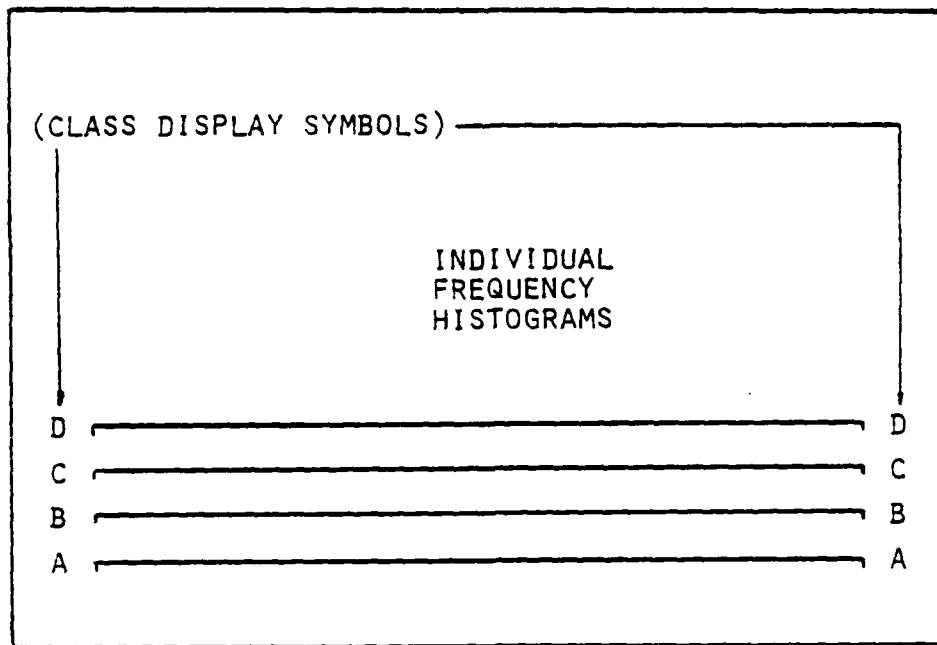
FUNCTIONAL ASPECT OF CLPARS -- 2
DISPLAY DESCRIPTIONS (one-space)

Following is a description of the information found on a one-space display.

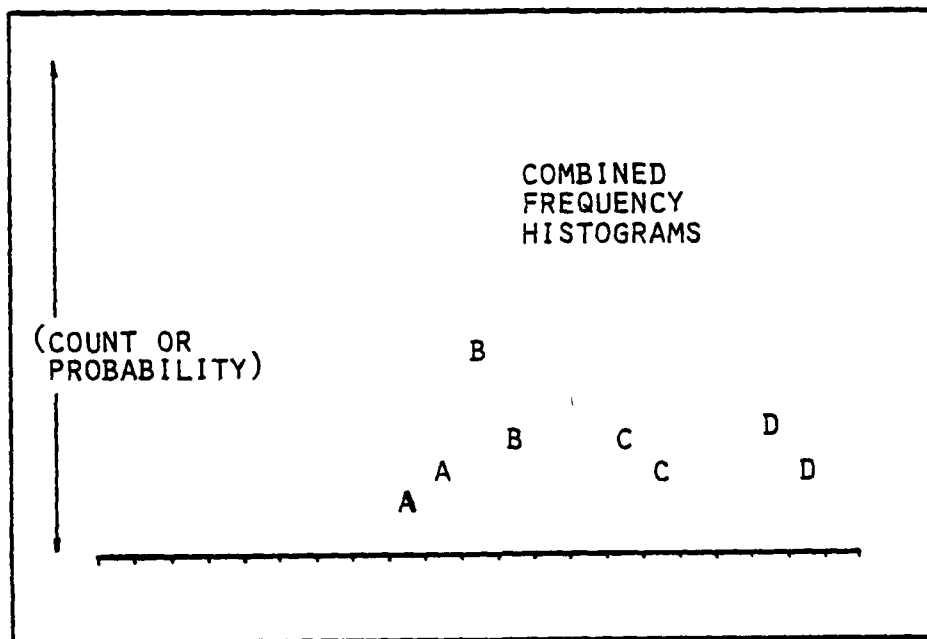
- (LL) BIN COUNT - The number of compartments (bins) into which the current data range has been divided.
- (LL) BIN SIZE - The width or size of a single bin (computed by dividing the data range by the number of bins found on the display).
- CLASS DISPLAY SYMBOL - Symbol displayed is used to represent a particular data class.
- (LR) CURRENT OPTION (OPTION NAME) - Command used to create the one-space display.
- (L) DATA CLASSES PRESENT - A list of display symbols of the classes found in the current data set (includes DISPLAY FLAG and DISPLAY FLAG SYMBOL).
- (LR) DATA SET (TREE NAME) (NODE NAME) - The current data set name, consisting of the current data tree and node name pair.
- (UR) DATE, TIME - The date and time the current display has been generated.
- DISPLAY FLAG - When an asterisk appears next to the class display symbol (found in the classes - present list), projected vectors from that particular class may be found on the display. If no asterisk appears, the vector from that class will not be found on the display.
- DISPLAY FLAG SYMBOL - A non-blank symbol appearing between the display flag and the class display symbol. The symbol is used to show a "grouping" of the classes present.

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (one-space)

- (LL) INTERACTION NOTES - Region reserved for terse prompts prior to expected graphics input.
- (UL) MEASUREMENTS USED - For coordinate projection, this entity represents the measurement used to create the current display. For other projections, this entity is the number of the projection vector used to create the display.
- (R) MENU - List of "suggested" commands to be used by operator.
- (C) PLOT DISPLAY REGION - Central area of the screen containing the projected data set display (see Figure 2-3 for plot examples)
- (UL) PLOT TYPE - The words "MACRO" or "MICRO" denoting which type of plot is being displayed.
- (UL) SCALE MODE - The words "GLOBAL" or "ZOOM" indicating that the entire data range appears on the current display (GLOBAL), or only a portion of the data range is viewable (ZOOM).
- (UL) SCALE TYPE - The words "PROBABILITIES" or "VECTOR COUNTS" showing that either the probability that the vectors will be located in a certain bin (i.e., the number of vectors in that bin divided by the total number of vectors in the data class), or that the number of vectors found in the bin is being used in scaling the current display.
- (LL) X-MINIMUM VALUE - Value of the left side of the data projection line found on the display (the low end of the current data range).
- (LL) X-MAXIMUM VALUE - Value of the right side of the data projection line found on the display (the high end of the current data range).



1-SPACE MACRO VIEW



1-SPACE MICRO VIEW

Figure 2-3 One-Space Macro and Micro Views

2.3.2 Two-Space Displays (SCATTER And CLUSTER) -

A two-space scatter plot is a two-dimensional representation of an N-space vector, with each vector located at its "natural" projection point on the screen.

A two-space cluster plot is a two-dimensional representation of an N-space vector, with each vector "forced" into a location within a grid. If one or more vectors from a single data class fall within the same grid location, the display symbol for that class is presented. If vectors from two or more data classes fall within a single grid location, an asterisk is displayed.

The actual presentation of the two-space cluster display is generally faster than the two-space scatter display, especially for a large data set. However, since each character displayed may represent one or more vectors, this display could be misleading in terms of vector density.

2.3.2.1 Two-Space Display Format -

The two-space display format is similar to the one-space display format with respect to the following entities; the data-classes-present list, plot type, measurements used, date, time, menu, current data set name, current option name, and a user interaction notes area (see Figure 2-4).

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (two-space)

Following is a description of the information found on a two-space display.

- | | |
|---------------------------------------|--|
| CLASS DISPLAY SYMBOL | - Displayed symbol is used to represent a particular data class. |
| (LR) CURRENT OPTION (OPTION NAME) | - Command used to create the two-space display. |
| (L) DATA CLASSES PRESENT | - A list of display symbols of the classes found in the current data set (includes DISPLAY FLAG and DISPLAY FLAG SYMBOL). |
| (LR) DATA SET (TREE NAME) (NODE NAME) | - The current data set name, consisting of the current data tree and node name pair. |
| (UR) DATE, TIME | - The data and time the current display has been generated. |
| DISPLAY FLAG | - When an asterisk appears next to the class display symbol (found in the classes - present list), projected vectors from that particular class may be found on the display. If no asterisk appears, the vectors from that class will not be found on the display. |
| DISPLAY FLAG SYMBOL | - A non-blank symbol appearing between the display flag and the class display symbol. The symbol is used to show a "grouping" of the classes present. |
| (LL) INTERACTION NOTES | - Region reserved for terse prompts prior to expected graphics input. |
| (UL) MEASUREMENTS USED | - For coordinate projections, the numbers represent the measurements used to create the current display. For other projections, the numbers represent the projection vectors used to create the display. |

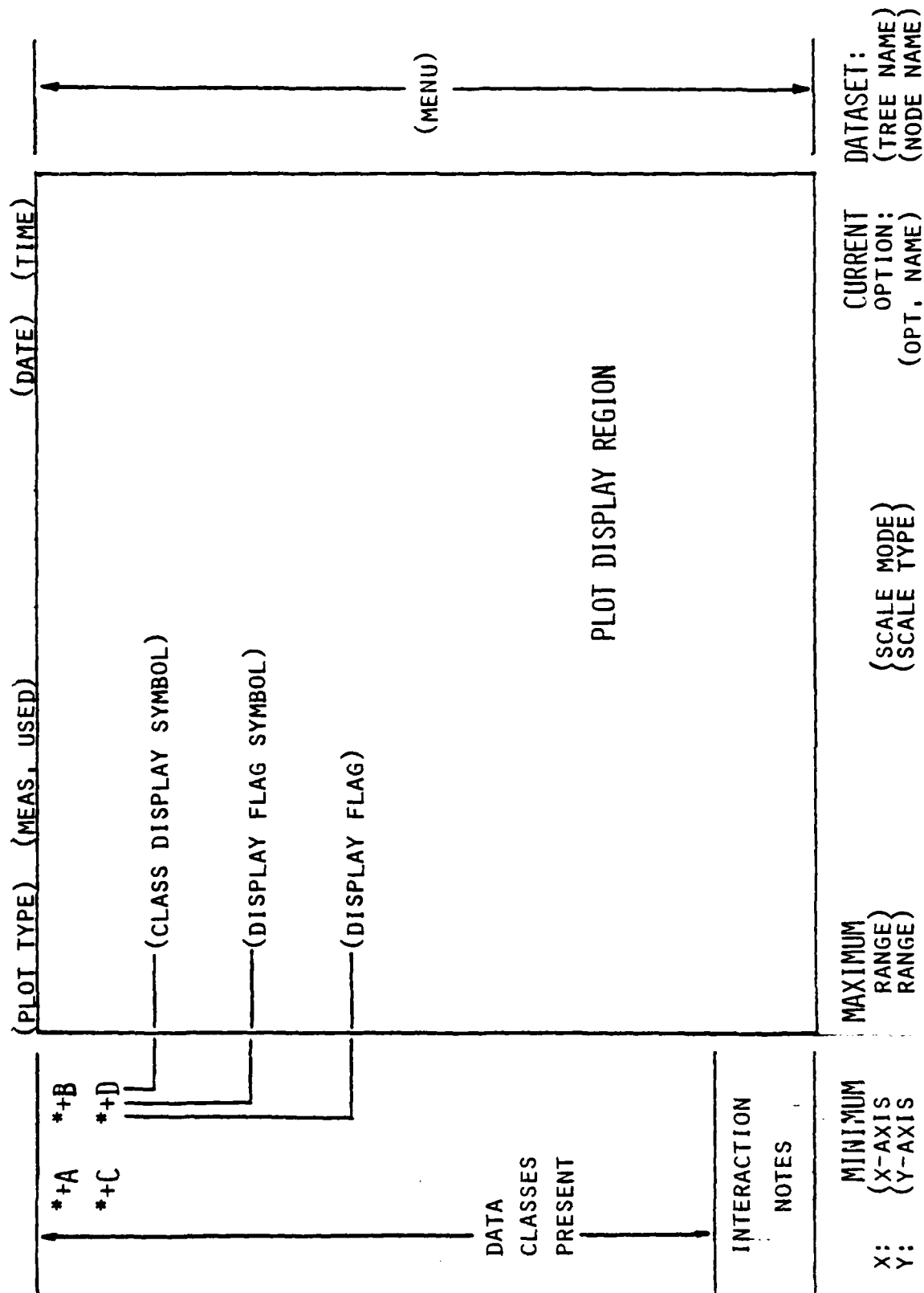
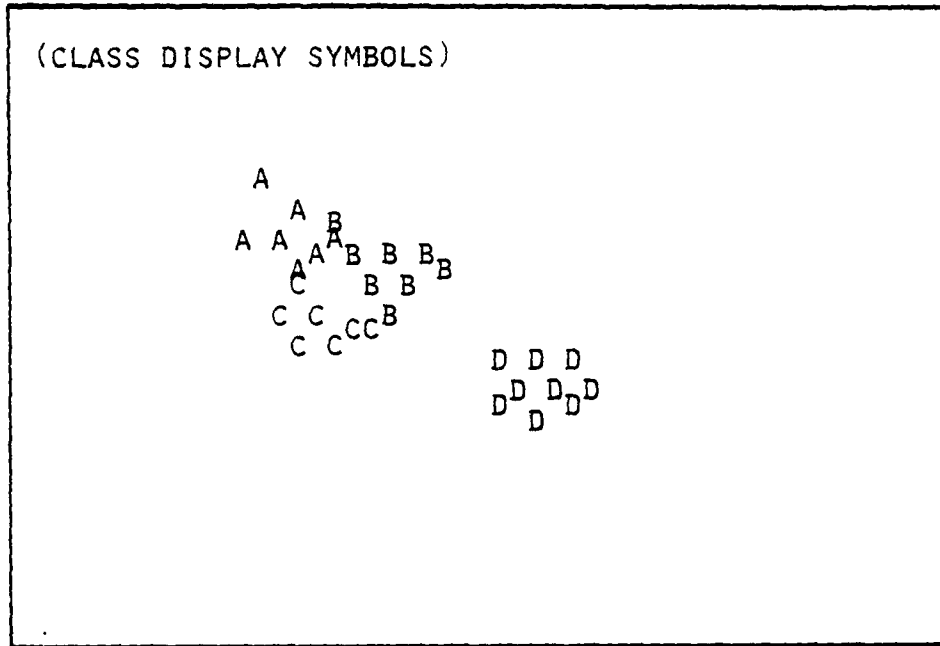


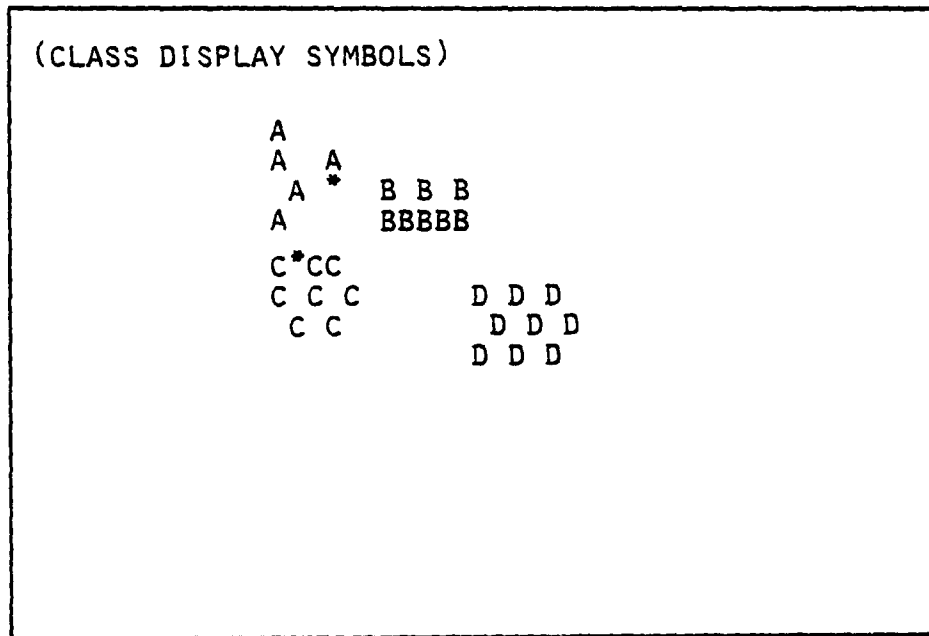
Figure 2-4 General Format of Two-Space Display

FUNCTIONAL ASPECT OF OLPAAS -- 2
DISPLAY DESCRIPTIONS (two-space)

- (R) MENU - List of "suggested" commands to be used by the operator.
- (C) PLOT DISPLAY REGION - Central area of the screen containing the projected data set vectors (see Figure 2-5 for plot examples).
- (UL) PLOT TYPE - The words "SCATTER" or "CLUSTER" denoting which type of plot is being displayed.
- (LC) SCALE MODE - The words "GLOBAL" or "ZOOM" indicating that the entire data range appears on the current display (GLOBAL), or only a portion of the data range is viewable (ZOOM).
- (LC) SCALE TYPE - The words "SQUARE" or "RECTANGULAR"; initially, all two-space projections have "SQUARE" scaling. This means that the value of the measurement units on both the x and y axes are equal. When scale "zooming" (obtaining a close-up view of a subsection of the original display) or a scale change occurs, the scaling becomes "rectangular", that is, the value of the measurement units on the x and y axes are no longer equal.
- (LL) X-AXIS RANGE - The minimum and maximum data values found on the first projection axis (the horizontal axis of projection).
- (LL) Y-AXIS RANGE - The minimum and maximum data values found on the second projection axis (the vertical axis of projection).



2-SPACE SCATTER PLOT



2-SPACE CLUSTER PLOT

Figure 2-5 Two-Space Scatter and Cluster Plots

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (two-space)

2.3.3 Confusion Matrix Displays -

(BETWEEN-GROUP, WITHIN-GROUP, OVERALL)

A "confusion matrix" is an ordered table of vector tallies used to show the results of partial or overall logical evaluation.

A "between-group" confusion matrix is generated after evaluating logic created via a one-space or two-space data partitionment (group logic). Each row in the matrix presents the evaluation results of an individual class. The left-most column of the matrix shows how many vectors belong to each data class. The remaining columns show the tally of vectors (from each class) belonging to a group (region). Class statistics (across a matrix row) and group statistics (down a matrix column) are presented showing the number and percentage of vectors correctly and incorrectly classified. Statistics are also presented for the total number of vectors evaluated.

NOTE

Since it may be desirable to redisplay the data set projections and the partitions upon which the evaluated logic is based, a "between-group" confusion matrix is not stored in a display file (unlike "within-group" confusion matrices). Consequently, for a "between-group" confusion matrix that contains more than one page of information, a terminal screen copying mechanism should be provided to record the multiple page display.

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (confusion matrix)

Figure 2-6 shows the general layout of a "between-group" confusion matrix. Each named area is described in greater detail in the following text. The information outside of the "blocked" diagram is described first, followed by "block" descriptions, according to their numbered order. An actual "between-group" confusion matrix display can be seen in Figure 2-7.

Following is a description of the information found on a between-group confusion matrix display.

CURRENT OPTION (OPTION NAME) - Command used to create the logic which in turn is used to create the between-group confusion matrix.

DATA SET (TREE NAME) (NODE NAME) - The current data set name, consisting of the current data tree and node name pair.

DATE TIME - The date and time the current display has been generated.

DISPLAY TYPE - The words "CONFUSION MATRIX (BTWN. GROUP LOGIC)" indicating the type of display.

MENU - List of "suggested" commands to be used by the operator.

-
1. REGIONS - Partitioned regions of one-space or two-space plots containing user specified data classes; one-space regions: "LEFT", "RIGHT", "MIDDLE", two space regions: "CONVEX (1)", "CONVEX (2)", "EXCESS".
 2. LOGIC NODE - Logic node numbers associated with user defined regions.

| (DISPLAY TYPE) | | (DATE) (TIME) | | (MENU) |
|--------------------------------------|-------------------------------------|---|---------------------------------------|---------------------------------------|
| 1 REGIONS | 2 LOGIC NODE | 3 DISPLAY SYMBOLS OF ASSOCIATED CLASSES | 4 LOGIC NAME | |
| 5 CLASS NAMES | 6 L O G I C (PARENT) | 7 N O D E S (CHILDREN) | 8 CLASS SUMS AND PERCENTAGES | |
| | VECTOR COUNTS | | | |
| 9 LOGIC NODE SUMS AND PERCENTAGES | 10 TOTAL SUMS AND PERCENTAGES | | | |
| CURRENT OPTION: (OPT. NAME) | | | | DATASET (TREE NAME) (NODE NAME) |

Figure 2-6 General Format of Between-Group Confusion Matrix

CONFUSION MATRIX (BTWN. GROUP LOGIC) DATE: 02-FEB-82 14:27:16

| REGION | LOGIC NODE | DISPLAY SYMBOLS OF ASSOCIATED CLASSES | LOGIC NAME- GRAINS | CDISPLAY CREATLOG CSCALE DBNDY DRAWBNDY PROJMN PRTIDX RDISPLAY REDRAW REPROJECT SCALZM SCALRET SELECT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---------------|--|--------------------|---|------------------|----------------------|------------------|--------|-----------|--------|----------------------|--------|--------|----------|--------|-----------|--------|---------|--------|---------------|-----------------|-----------------|--------------------|------------------|--------------------|------------------|-----|----|---|----|----|---|-------|-----|--|--|--|--|--|--|--|------|----|---|----|----|---|------|------|--|--|--|--|--|--|--|------|----|---|----|----|---|-------|-----|--|--|--|--|--|--|--|-------|----|----|---|---|---|-------|-----|--|--|--|--|--|--|--|------|----|---|---|---|----|-------|-----|--|--|--|--|--|--|--|------|----|---|---|---|----|-------|-----|--|--|--|--|--|--|--|-----|----|---|----|----|---|-------|-----|--|--|--|--|--|--|--|-------|-----|----|----|-----|---|------|-----|--|--|--|--|--|--|--|--------------------|--|----|----|----|--|--|--|--|--|--|--|--|--|--|-------------------|--|-------|-------|------|--|--|--|--|--|--|--|--|--|--|--|--|-----|-----|-----|--|--|--|--|--|--|--|--|--|--|
| CONVEX (1) | 2 | w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CONVEX (2) | 3 | src | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EXCESS | 4 | ac | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th rowspan="2">CLASS NAMES</th> <th colspan="3">L O G I C</th> <th colspan="2">N O D E S</th> <th colspan="2">SUMS AND PERCENTAGES</th> <th rowspan="2">PRTIDX</th> <th rowspan="2">RDISPLAY</th> <th rowspan="2">REDRAW</th> <th rowspan="2">REPROJECT</th> <th rowspan="2">SCALZM</th> <th rowspan="2">SCALRET</th> <th rowspan="2">SELECT</th> </tr> <tr> <th>(PARENT) 1</th> <th>(CHILDREN) 2</th> <th>(CHILDREN) 3</th> <th>(CORRECT) COUNT</th> <th>(ERROR) COUNT</th> <th>(CORRECT) PRCNT</th> <th>(ERROR) PRCNT</th> </tr> </thead> <tbody> <tr> <td>soy</td> <td>19</td> <td>0</td> <td>19</td> <td>19</td> <td>0</td> <td>100.0</td> <td>0.0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>corn</td> <td>18</td> <td>0</td> <td>16</td> <td>16</td> <td>2</td> <td>88.9</td> <td>11.1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>oats</td> <td>20</td> <td>0</td> <td>20</td> <td>20</td> <td>0</td> <td>100.0</td> <td>0.0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>wheat</td> <td>18</td> <td>18</td> <td>0</td> <td>0</td> <td>0</td> <td>100.0</td> <td>0.0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>clov</td> <td>19</td> <td>0</td> <td>0</td> <td>0</td> <td>19</td> <td>100.0</td> <td>0.0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>alfa</td> <td>17</td> <td>0</td> <td>0</td> <td>0</td> <td>17</td> <td>100.0</td> <td>0.0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>rye</td> <td>18</td> <td>0</td> <td>18</td> <td>18</td> <td>0</td> <td>100.0</td> <td>0.0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>TOTAL</td> <td>129</td> <td>18</td> <td>73</td> <td>127</td> <td>2</td> <td>98.4</td> <td>1.6</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>CORRECT (PRCNT)</td> <td></td> <td>18</td> <td>73</td> <td>38</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>ERRORS (PRCNT)</td> <td></td> <td>100.0</td> <td>100.0</td> <td>94.7</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td>0.0</td> <td>0.0</td> <td>5.3</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table> | | | | | CLASS NAMES | L O G I C | | | N O D E S | | SUMS AND PERCENTAGES | | PRTIDX | RDISPLAY | REDRAW | REPROJECT | SCALZM | SCALRET | SELECT | (PARENT) 1 | (CHILDREN) 2 | (CHILDREN) 3 | (CORRECT) COUNT | (ERROR) COUNT | (CORRECT) PRCNT | (ERROR) PRCNT | soy | 19 | 0 | 19 | 19 | 0 | 100.0 | 0.0 | | | | | | | | corn | 18 | 0 | 16 | 16 | 2 | 88.9 | 11.1 | | | | | | | | oats | 20 | 0 | 20 | 20 | 0 | 100.0 | 0.0 | | | | | | | | wheat | 18 | 18 | 0 | 0 | 0 | 100.0 | 0.0 | | | | | | | | clov | 19 | 0 | 0 | 0 | 19 | 100.0 | 0.0 | | | | | | | | alfa | 17 | 0 | 0 | 0 | 17 | 100.0 | 0.0 | | | | | | | | rye | 18 | 0 | 18 | 18 | 0 | 100.0 | 0.0 | | | | | | | | TOTAL | 129 | 18 | 73 | 127 | 2 | 98.4 | 1.6 | | | | | | | | CORRECT (PRCNT) | | 18 | 73 | 38 | | | | | | | | | | | ERRORS (PRCNT) | | 100.0 | 100.0 | 94.7 | | | | | | | | | | | | | 0.0 | 0.0 | 5.3 | | | | | | | | | | |
| CLASS NAMES | L O G I C | | | N O D E S | | SUMS AND PERCENTAGES | | PRTIDX | RDISPLAY | REDRAW | REPROJECT | SCALZM | | | | | | | | SCALRET | SELECT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | (PARENT) 1 | (CHILDREN) 2 | (CHILDREN) 3 | (CORRECT) COUNT | (ERROR) COUNT | (CORRECT) PRCNT | (ERROR) PRCNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| soy | 19 | 0 | 19 | 19 | 0 | 100.0 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| corn | 18 | 0 | 16 | 16 | 2 | 88.9 | 11.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| oats | 20 | 0 | 20 | 20 | 0 | 100.0 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| wheat | 18 | 18 | 0 | 0 | 0 | 100.0 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| clov | 19 | 0 | 0 | 0 | 19 | 100.0 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| alfa | 17 | 0 | 0 | 0 | 17 | 100.0 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| rye | 18 | 0 | 18 | 18 | 0 | 100.0 | 0.0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TOTAL | 129 | 18 | 73 | 127 | 2 | 98.4 | 1.6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CORRECT (PRCNT) | | 18 | 73 | 38 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ERRORS (PRCNT) | | 100.0 | 100.0 | 94.7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 0.0 | 0.0 | 5.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

CURRENT
OPTION:
L2EIGU

DATASET:
GRAINTST

Figure 2-7 Between-Group Confusion Matrix Example

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (confusion matrix)

3. DISPLAY SYMBOLS - Display symbols of data classes assigned
 OF ASSOCIATED to a given region.
 CLASSES
4. LOGIC NAME - Current logic tree name.
5. CLASS NAMES - Names of the data classes present at the
 node for which logic has been created.
6. LOGIC NODES
 (PARENT) - Node at which logic has been created.
 (CHILDREN) - Nodes associated with user defined
 regions.
7. VECTOR
 COUNTS - Number of vectors found at each logic
 node after evaluation.
8. CLASS SUMS - The counts and percentages of the number
 AND PERCENTAGES of correctly and incorrectly classified
 vectors of a given class.
9. LOGIC NODE
 SUMS AND - The counts and percentages of the
 PERCENTAGES number of correctly and incorrectly
 classified vectors at a given logic node.
10. TOTAL SUMS - The total count and percentage of
 AND PERCENTAGES correctly and incorrectly classified
 vectors.

A "within-group" confusion matrix is generated after evaluating logic created by pairwise discriminant logic or nearest mean vector logic. The main section of the display consists of a numeric matrix in which the column labels correspond to the data classes on which logic was designed (assigned classes). The row labels are associated with the classes of the data set being evaluated (true classes). Any element of this matrix is the number

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (confusion matrix)

of vectors, from a given data class, assigned to a particular logic node. Following the matrix are various totals and percentages associated with the rows (true classes) of the matrix. These tallies represent the number and percentage of vectors classified correctly and incorrectly. Included is a count and percentage of vectors which may have been rejected during classification.

An "overall" confusion matrix is produced after evaluating a completed logic tree. The format of this confusion matrix is identical to that of the "within-group" confusion matrix. Figure 2-8 shows the general format of a "within-group" confusion matrix. The menu, current option, and current data set are not shown, but are in the same position as in the other displays. Following is a detailed description of the "blocked" entities found in the figure. An actual "within-group" display can be seen in Figure 2-9, following the general format Figure.

1. LOGIC IDENTIFICATION - Contains logic type (e.g. nearest mean vector), logic tree name, design data set name and dimensionality.
2. ASSIGNED CLASS NAMES - Names of the data classes on which logic was designed.
3. TRUE CLASS NAMES - Names of the data class evaluated against the current logic.
4. VECTOR COUNTS - Number of vectors from a true class determined to belong to a particular assigned class (design data set class).

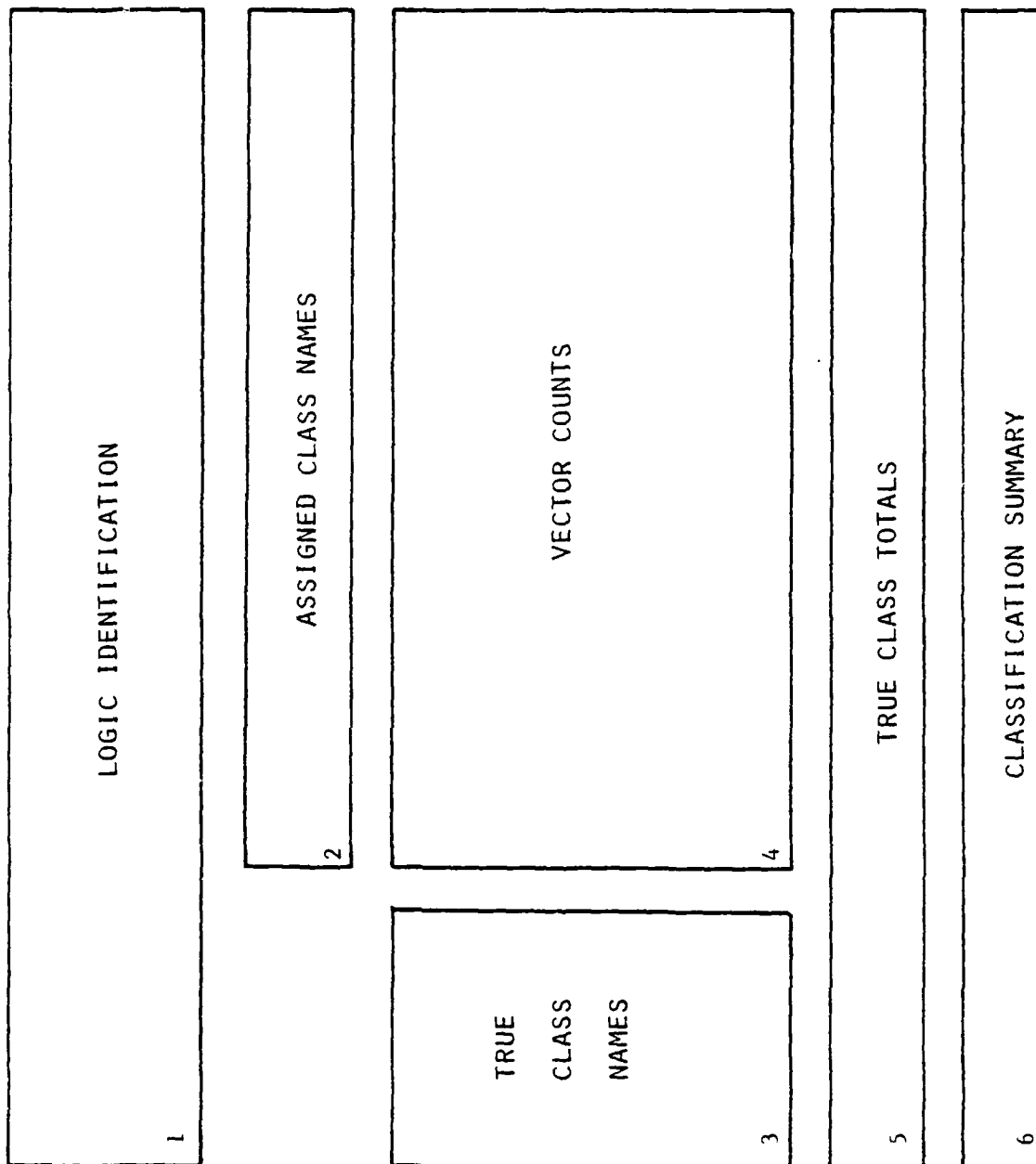


Figure 2-8 General Format of Within-Group Confusion Matrix

PARTIAL PAIRWISE EVALUATION FOR LOGIC MODE 1
 LOGIC NAME: GRAINS DESIGN DATA SET: GRAINTST (***)
 DIMENSIONALITY - 7

ASSIGNED CLASSES

| | soy | corn | oats | weat | Clov | alfa | rye |
|------|-----|------|------|------|------|------|-----|
| soy | 19 | 0 | 0 | 0 | 0 | 0 | 0 |
| corn | 0 | 18 | 0 | 0 | 0 | 0 | 0 |
| oats | 0 | 0 | 20 | 0 | 0 | 0 | 0 |
| weat | 0 | 0 | 0 | 18 | 0 | 0 | 0 |
| Clov | 0 | 0 | 0 | 0 | 18 | 1 | 0 |
| alfa | 0 | 0 | 0 | 0 | 0 | 17 | 0 |
| rye | 0 | 0 | 1 | 0 | 0 | 0 | 17 |

TRUE CLASSES

| | soy | corn | oats | weat | Clov | alfa | rye |
|-------|-------|-------|-------|-------|------|-------|------|
| TOTAL | 19 | 18 | 20 | 18 | 19 | 17 | 18 |
| CORR | 19 | 18 | 20 | 18 | 18 | 17 | 17 |
| PRCT | 100.0 | 100.0 | 100.0 | 100.0 | 94.7 | 100.0 | 94.4 |
| EROR | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| PRCT | 0.0 | 0.0 | 0.0 | 0.0 | 5.3 | 0.0 | 5.6 |
| REJT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PRCT | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

TOTAL NUMBER OF VECTORS = 129
 OVERALL CORRECT 127 FOR 98.45 PRCT
 OVERALL ERROR 2 FOR 1.55 PRCT
 OVERALL REJECT 0 FOR 0.00 PRCT

FISHMOD
 OPTIMLMOD
 PRTCM
 RDISPLAY
 SUMMACH
 THRESHMOD

DATASET:
 GRAINTST

CURRENT
 OPTION:
 PWEVAL

Figure 2-9 Within-Group Confusion Matrix Example

FUNCTIONAL ASPECT OF CLPARS -- 2
DISPLAY DESCRIPTIONS (confusion matrix)

5. TRUE CLASS TOTALS - The counts and percentages of the number of correctly and incorrectly classified vectors of each evaluated data class. A count and percentage of the number of vectors rejected from each class is also presented.
6. CLASSIFICATION SUMMARY - The total number of vectors evaluated and the overall number and percentage of vectors correctly and incorrectly classified, along with the total number and percentage of vectors rejected, appears here.

2.3.4 Rank Order Displays -

A rank order display gives a measure of the discriminating power of a set of measurements. There are five types of rank order displays:

- (1) overall ranking of measurements
- (2) measurement ranking for a single class
- (3) measurement ranking for a class pair
- (4) class ranking for a single measurement
- (5) class pair ranking of a single measurement

Following is a description of the information found on each type of rank order display (See Figure 2-10). There are two categories of information described here - general information and ranking information.

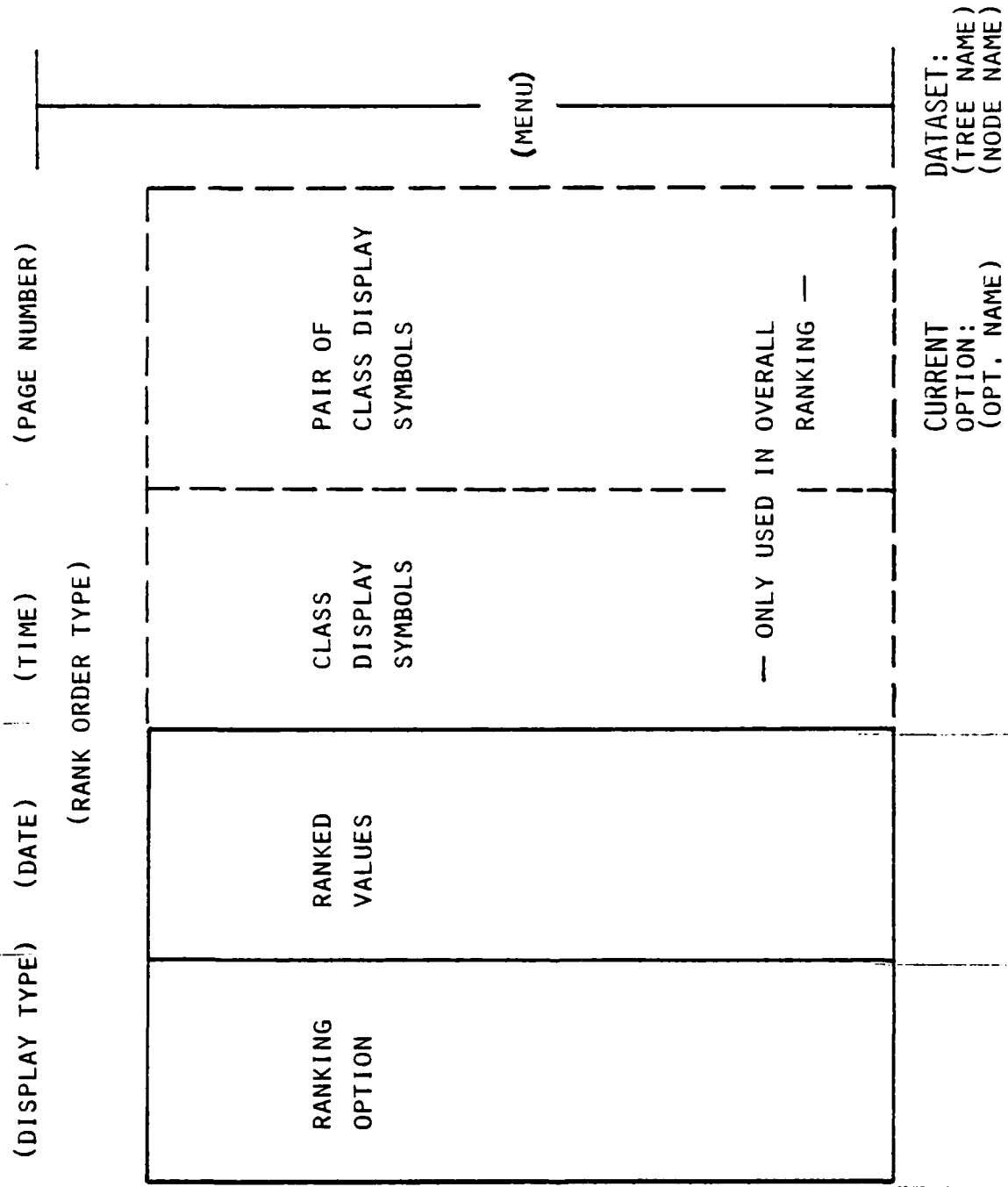


Figure 2-10 General Format of Rank Order Display

FUNCTIONAL ASPECT OF DISPLAYS -- 2
DISPLAY DESCRIPTIONS (rank order)

GENERAL INFORMATION

- | | |
|---------------------------------------|--|
| CLASS DISPLAY SYMBOL | - Symbol displayed is used to represent a particular data class. |
| (LR) CURRENT OPTION (OPTION NAME) | - Command used to create the rank order display. |
| (LR) DATA SET (TREE NAME) (NODE NAME) | - The current data set name, consisting of the current data tree and node name pair. |
| (UC) DATE, TIME | - The date and time the current display has been generated. |
| (UL) DISPLAY TYPE | - The words "A RANK ORDER DISPLAY" identifying the type of display. |
| MEASUREMENT NUMBER | - Position of an element in a data vector. |
| (R) MENU | - List of "Suggested" commands to be used by the operator. |
| (UR) PAGE NUMBER | - If the display consists of more than one page, the word "PAGE" is followed by the page number. |
| PAIR OF CLASS DISPLAY SYMBOLS | - Two class display symbols separated by a slash (/); used to represent a pair of data classes (class pair). |
| (UC) RANK ORDER TYPE | - One of the following five phrases will identify the type of rank order display (Note: "C" represents a class display symbol, "C/P" represents a pair of class display symbols, and "X" represents a measurement number.) |

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (rank order)

- (1) "AN OVERALL RANKING"
- (2) "A MEASUREMENT RANKING OF CLASS C"
- (3) "A MEASUREMENT RANKING OF CLASS
PAIR C/P"
- (4) "A CLASS RANKING OF MEASUREMENT X"
- (5) "A CLASS PAIR RANKING OF
MEASUREMENT X"

2.3.4.1 Ranking Information -

A rank order display consists of at most four columns which will be described in the following text in order of their occurrence from left to right on the display.

- | | |
|------------------------------|---|
| RANKING OPTION (Column 1) | - The first column in the rank order display represents the item being ranked; measurement numbers, class display symbols, or pair of class display symbols. |
| (1) Measurement number | - Measurement numbers are ranked according to their ability (power) to discriminate (separate) either (a) all classes (overall), (b) a given class from all other classes, or (c) one class from another. These three types of measurement rankings correspond to the first three types of rank order displays previously mentioned. In a ranking of measurement numbers, measurements which have the most discriminating power, appear at the top of the list. An asterisk (*) preceeding a measurement number indicates the measurement has been selected for a data tree transformation. |
| (2) Class display symbol | - A ranking of classes based on the power a given measurement has to separate them from all other classes. Those classes that are best separated from all other classes by the particular measurement occur at the top of the list. |
| (3) Pair of class display | - A ranking of pairs of classes based on the power a given measurement has to separate |

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (rank order)

symbols the first class in the pair from the second class in the pair. Those pairs of classes that are best separated from each other by the given measurement are found at the top of the list.

RANKED VALUES - Values that provide an estimate of the
(Column 2) ability of one or more measurements to separate either

- (a) all classes
- (b) a particular class from all other classes
- (c) one particular class from another class.

ADDITIONAL INFORMATION
(FOR AN OVERALL RANKING ONLY)

Class display - Each symbol corresponds to a measurement
symbol number found in the first column (same
(Column 3) row) of the display. The class display
symbol represents the class that is best
separated from all other classes by this
corresponding measurement number.

Pair of class - Each pair of symbols corresponds to a
symbols measurement number in the first column
(Column 4) (same row) of the display. The pair of
class display symbols represents the pair
of classes that is best separated from
each other by this corresponding
measurement number.

2.3.5 Data Tree Structural Displays -

OLPARS vector data is represented as a hierarchical tree (see the section on "OLPARS Data Representation"). A structural picture of an OLPARS data tree may be obtained at any time (see DRAWTREE command description). The data tree display consists of node names, the number of vectors at each node, and interconnecting

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (data tree)

structural lines showing the relationship between nodes.

Following is a description of the information found on an OLPARS data tree display (see Figure 2-11). An actual data tree may be seen in Figure 2-12.

- | | | |
|------|--|---|
| (LR) | CURRENT OPTION (OPTION NAME) | - OLPARS command (see Section 2.1.2) |
| (UL) | DATA TREE NAME | - The name of the data tree being displayed. |
| (LR) | DATA SET (TREE NAME) (NODE NAME) | - The current data set name, consisting of the current data tree and node name pair. |
| (UC) | DATE, TIME | - The date and time the current display has been generated. |
| (UL) | DISPLAY TYPE | - The words "OLPARS DATA TREE" identifying the type of display. |
| (R) | MENU | - List of "suggested" commands to be used by the operator. |
| | NODE NAME | - The name of a class or group of vectors ("*****" represents the senior node of a data tree) |
| | MEASUREMENT COUNT | - The dimensionality of the data tree being displayed (the number of measurements per vector). |
| | PAGE_OF_ | - If all nodes to be displayed do not fit on one page, the number of the page displayed and the total number of pages for the display is shown. |
| | VECTOR COUNT | - The number of data vectors at a lowest (leaf) node. |

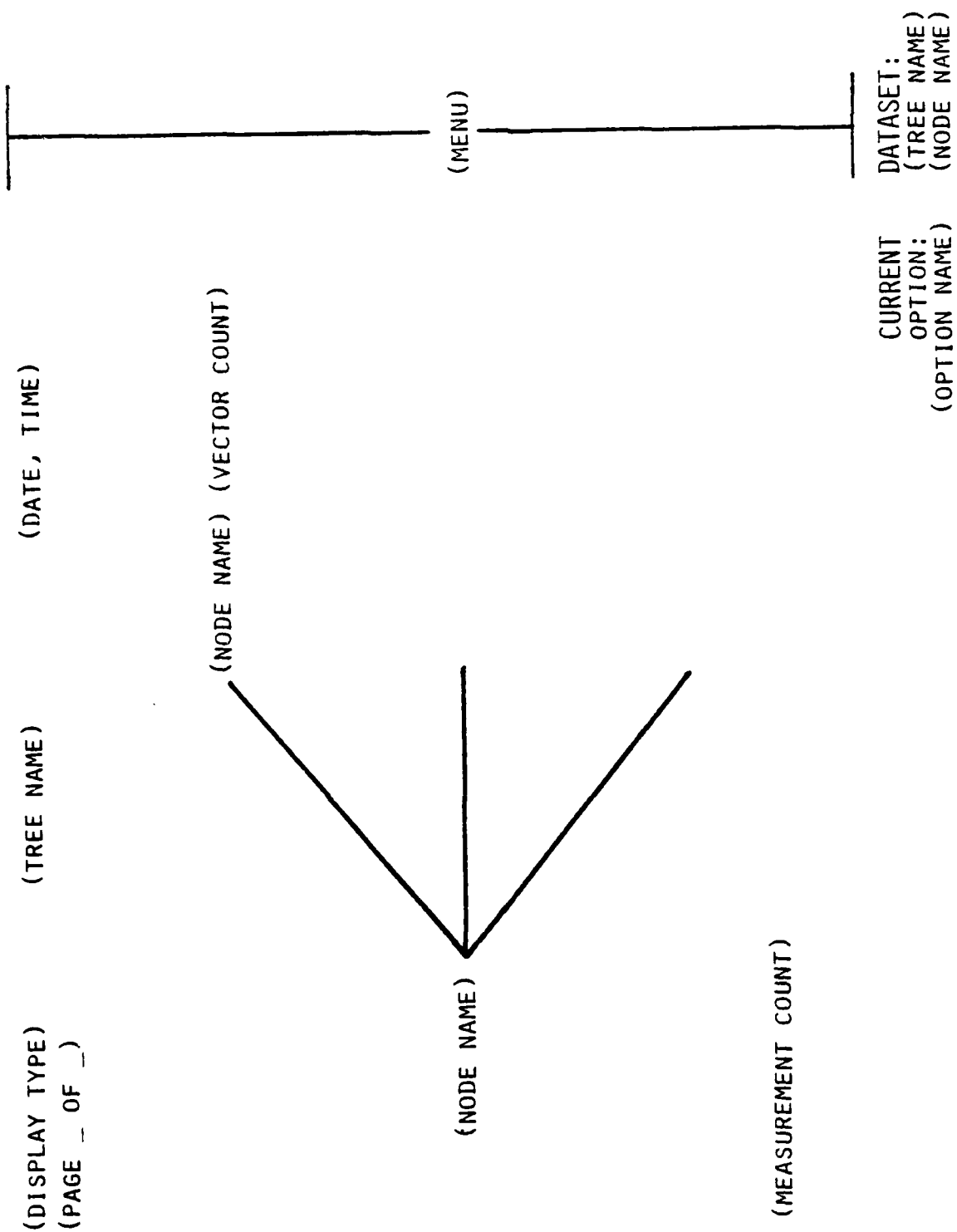
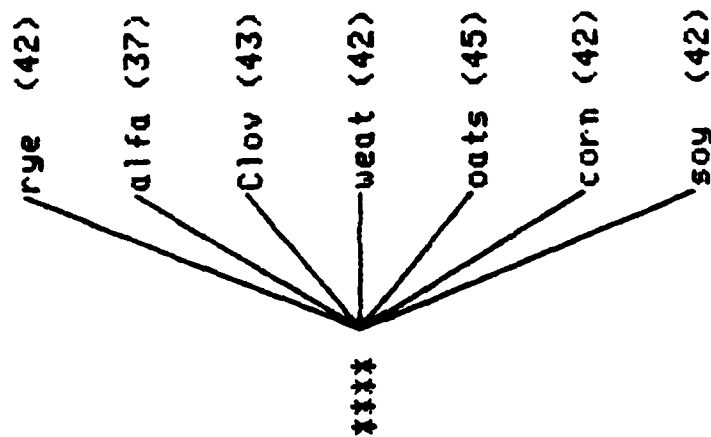


Figure 2-11 General Format of Data Tree Display

OLPARS DATA TREE 'grains' DATE: 18-JAN-82 13:37:40 ANYTHING



NUMBER OF MEASUREMENTS PER VECTOR 12

CURRENT
OPTION:
ANYTHING

DATASET:
grains

Figure 2-12 Data Tree Display Example

2.3.6 Logic Tree Structural Displays -

A display which gives a structural picture of an OLPARS logic tree may be obtained at any stage of logic development (see DRAWLOG). The logic tree display consists of subdivided rectangles (boxes) and interconnecting structural lines. Each box represents a node in the logic tree. Information displayed in each logic node consists of the logic node number (node identity), the type of logic at the node, and the classes present at the node. The interconnecting lines illustrate the relationship between nodes.

Following is a description of the information found on an OLPARS logic tree display (see Figure 2-13). Information from two categories will be described here - general information and logic node information. Logic node information is located inside the subdivided rectangles on the display.

GENERAL INFORMATION

- | | | |
|------|-------------------------|--|
| (LL) | CLASS COUNT | - The number of classes in the design data set. This is also the number of classes at the senior node of the logic tree. |
| | CLASS DISPLAY SYMBOL | - Symbol display is used to represent a particular data class |
| (UL) | CLASSES PRESENT | - A list of the class display symbols of the classes in the design data set. |

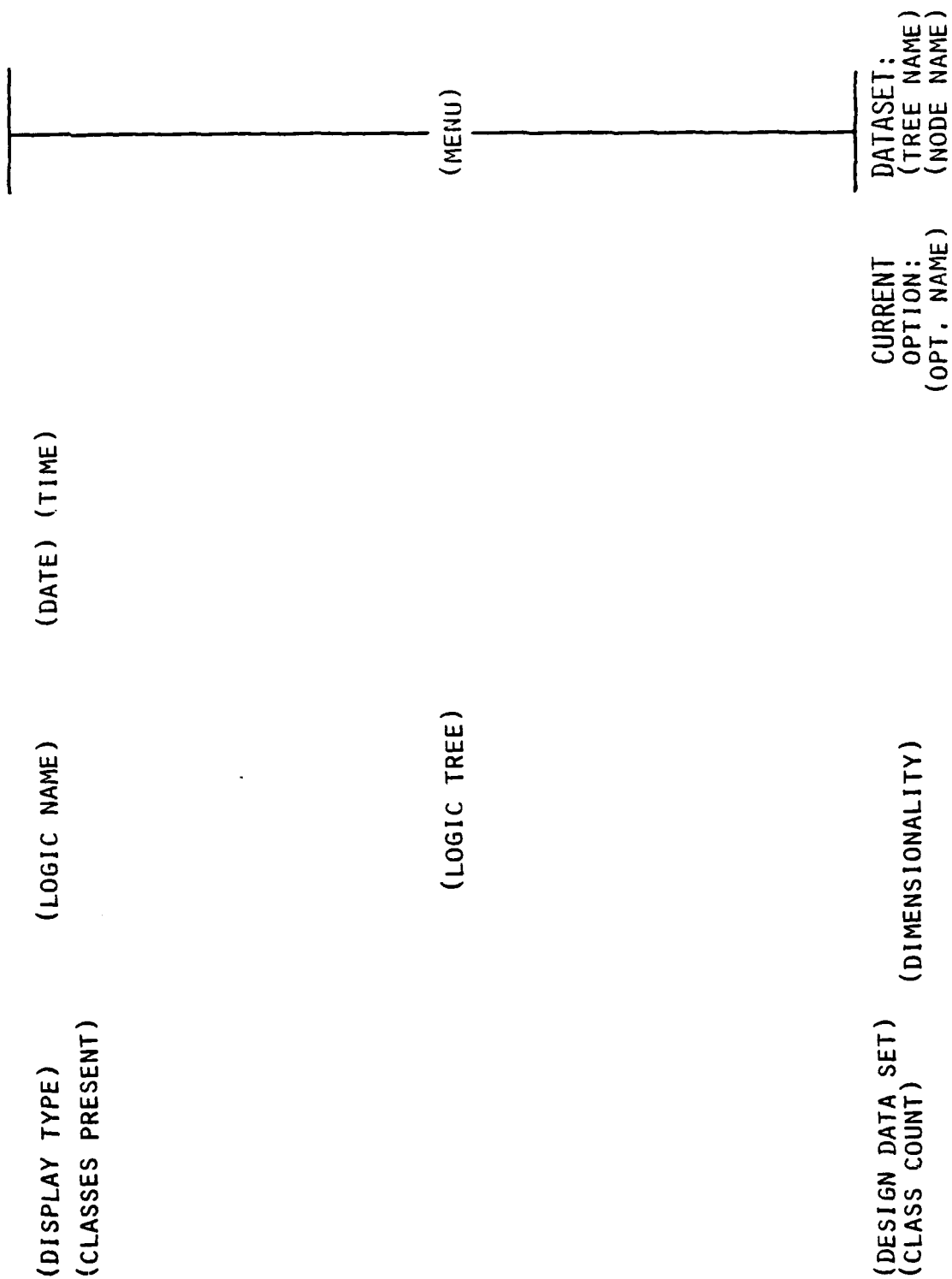


Figure 2-13 General Format of Logic Tree Display

FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (logic tree)

- (LR) CURRENT OPTION (OPTION NAME) - OLPARS command (see Section 2.1.2)
- (LA) DATA SET (TREE NAME) (NODE NAME) - The current data set name, consisting of the current data tree and node name pair.
- (JC) DATE, TIME - The date and time the current display has been generated.
- (LL) DESIGN DATA SET - The tree name, node name pair of the data set used to create the logic.
- (LL) DIMENSIONALITY - The dimensionality of the design data set vectors.
- (UL) DISPLAY TYPE - The words "OLPARS LOGIC TREE" identifying the type of display.
- (UL) LOGIC NAME - The name of the logic tree being displayed.
- (C) LOGIC TREE - Structural display of an OLPARS logic tree.
- (R) MENU - List of "suggested" commands to be used by the operator.

2.3.6.1 Logic Node Information -

Figure 2-14 identifies the regions of the subdivided rectangle which represents an OLPARS logic node. Following is a description of each region.

- CLASS(ES) PRESENT - If the logic node is incomplete, the class display symbols of the design data set classes residing at the logic node are shown. If not all class display symbols

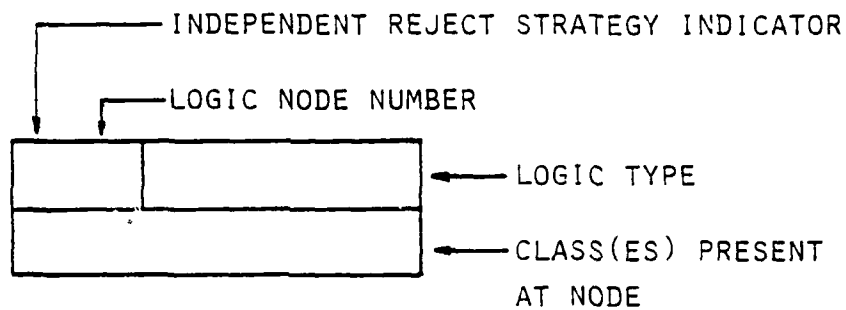
FUNCTIONAL ASPECT OF OLPARS -- 2
DISPLAY DESCRIPTIONS (logic tree)

fit in the box, those that fit are shown followed by "...". (indicating there are more). If the logic node is complete, there is only one class present, and its name appears in this portion of the box. For reject nodes, this region is empty.

| | |
|---|--|
| INDEPENDENT REJECT STRATEGY INDICATOR | - If there is an independent reject strategy at the logic node, an asterisk (*) appears in this location; otherwise this location is left blank. |
| LOGIC NODE NUMBER | - Identity of the logic node being presented. |
| LOGIC TYPE | - If logic has been designed at the node, the name of the OLPARS command creating the logic appears in this location. For logic nodes with no logic and more than one class present (incomplete logic nodes), the character string "INCMPLT" appears. For Reject nodes and logic nodes with only one class present (completed logic nodes), this region remains empty. |

Six examples of logic nodes, as they would appear on a display, are given in Figure 2-14 and are described in the following text. An actual logic tree display can be seen in Figure 2-15.

1. The command "L2EIGV" created logic at this node. All classes in the design data set are present ("...") indicates that not all class display symbols fit in the box). Note, when the logic node number is "1", the node is called the "senior node" of the logic tree.
2. The command "NMV" created logic at this node. Two classes are present. Their class display symbols are "A" and "B". The asterisk (*) indicates that there is an independent reject strategy at this node.
3. This logic node is incomplete, that is, no logic has been designed at this node, and there is more than one class present at the node.



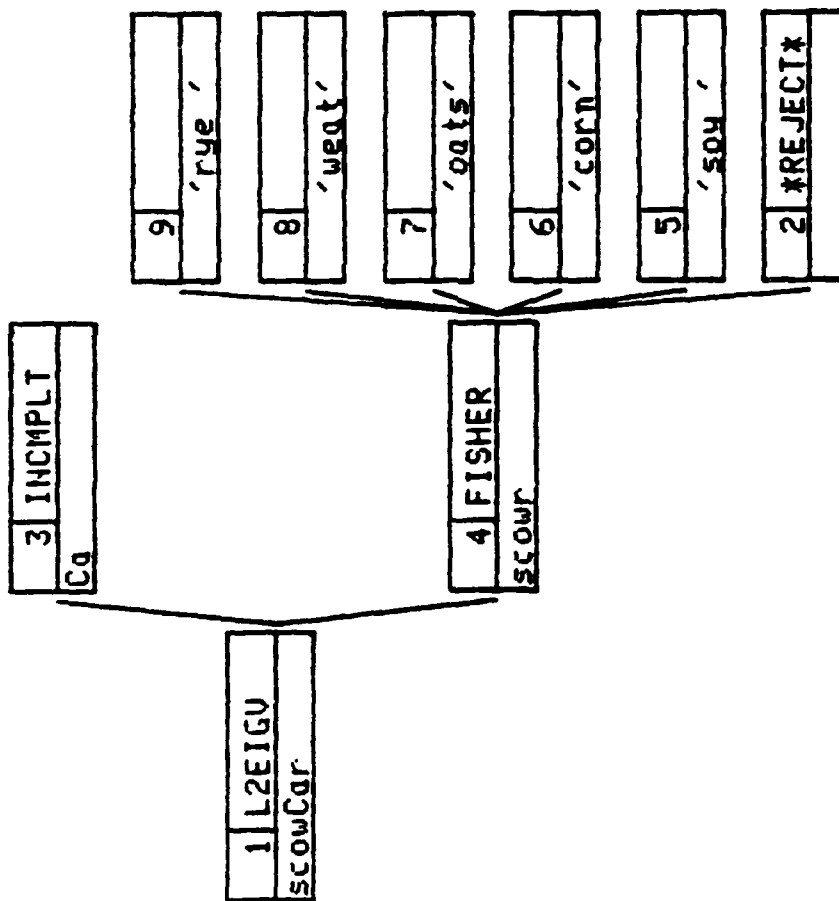
| | | |
|--|---|--|
| <div> <div>1 L2EIGV</div> <div>ABCDEFGG...</div> </div> <p>1</p> | <div> <div>*2 NMV</div> <div>AB</div> </div> <p>2</p> | <div> <div>3 INCMPLT</div> <div>CDEFGHI...</div> </div> <p>3</p> |
| <div> <div>4 *REJECT*</div> <div></div> </div> <p>4</p> | <div> <div>5 </div> <div>'ANOD'</div> </div> <p>5</p> | <div> <div>*6 </div> <div>'BNOD'</div> </div> <p>6</p> |

Figure 2-14 General Format of Logic Tree Nodes and Logic Node Examples

OLPARS LOGIC TREE 'GRAINS'
 CLASSES PRESENT: scowCar

DATE: 02-FEB-82 15:03:11

FISHMOD
 OPTIMLMOD
 PWEVAL
 THRESHMOD



DESIGN DATA SET: GRAINTST (***)
 CLASS COUNT: 7 DIMENSIONALITY: 7

CURRENT
 OPTION:
 FISHER

DATASET:
 GRAINTST

Figure 2-15 Logic Tree Display Example

FUNCTIONAL ASPECT OF CLPARS -- 2
DISPLAY DESCRIPTIONS (logic tree)

4. This is an example of a reject node. Note, the "class(es) present" portion of the box is empty.
5. This logic node is considered a completed logic node. There is one class present and its name is "ANCD".
6. This is another example of a completed logic node. There is one class present and its name is "BNOD". Note, an independent reject strategy exists at this node.

SECTION 3

USING OLPARS

3.0 YOUR'RE NEW, SO WHAT DO YOU DO?

This Section introduces a researcher to the usage of the analytic tool called OLPARS. New users are shown how to access OLPARS, and are given an initial view on how to approach analysis of their data* (because there is more than one method of analysis). As user's become more sophisticated, they will develop their own analytic methods. Use Appendix A for a mathematical reference.

3.1 NOTES ON DATA COLLECTION

Data collected for OLPARS must have the following characteristics: (1) It must be representative of the real world, (2) there must be a sufficient amount so that results obtained will be statistically significant (see Section 2.2) and (3) It must be in the form of vectors. Another factor to be considered is if the sample data being used to design a classifier is representative of the entire time period that the classifier will be used. For example, will a classifier developed using crop data collected in the spring be useful for classifying crop data collected in the

* The data set we will use to illustrate various points throughout this manual is named GRAINS. The features in the GRAINS data set are measurements of energy reflectivity of various crops. Each measurement represents a certain portion of the energy in the infra-red spectrum.

fall. In such a case, it may be necessary to add the dimension "time collected" to the data.

3.2 IMPORTANCE OF FEATURE EXTRACTION

As mentioned in the introduction of this manual, pattern recognition consists of feature extraction, and pattern classification. First and foremost, emphasis should be placed on feature extraction. It is important for you, the analyst, to know what each measurement in the data set you have collected represents.** Some measurements may be closely related to each other (the length of a side of a rectangle in relation to the area of the rectangle), while others may have no connection, whatsoever (the color of a human eye, in relation to the weight of a person). A feature can be an environmental "raw" measurement (weight of an object) or a statistic obtained from the "raw" measurements of the environment (number of peaks in a wave form).

During structure analysis or measurement evaluation, you may find that the features used to represent the objects being classified distinguish all but two of the objects. Another feature may have to be picked from the environment to separate these problem objects. The analysis and feature extraction process may have to iterate several times.

As can be seen in the following story, it is important to

** An example of feature extraction for the classification of hand printed numerals appears in [2].

obtain the proper attributes for recognizing an object in its environment.

"There were two farmers who had adjoining fields separated by a stonefence. Both farmers had a horse which they would let graze in their respective fields. Every evening when they would go to bring their horses in for the night, they found that one of the horses would have gotten across into the neighboring field to be with the other horse. Then the farmers would have to determine which horse was theirs. So they decided to tie a ribbon on one of the horses tails. However, when they located the horses together the following day, the ribbon was nowhere in sight (It had been torn off the horse by some nearby underbrush. This time, they decided to take a notch out of one of the horses ears to make it distinguishable from the other horse. The following day, when the horses were found, the farmers saw that both horses had notches in their ears. Sometime during that day the unmarked horse had gotten its ear caught in some barbed-wired fencing, which had torn a notch in its ear. The farmers were beginning to understand that they needed a tell-tale sign that could not be altered by the horses' day-to-day encounters. So they decided to measure the height of each of the horses. They used their hands as a measuring device. The first horse measured was fourteen hands high. Then the other horse was measured, and sure enough, the farmers found that the white horse was one-hand taller than the black horse."

3.3 HELLO OLPARS

If you have not used OLPARS before, you won't have an OLPARS directory properly set up for your use. You should see the local OLPARS manager/installer. The manager has to create some special files for you and should show you how to get access to the OLPAR system.

Making yourself known to OLPARS (better known as "logging in") may vary from system to system. You may have to specify a username, password, terminal type, etc. This information should be available from the OLPAR system manager.

USING OLPARS -- 3 CREATING A DATA TREE

3.4 CREATING A DATA TREE

Once you have "logged in", you will be ready to do some work. If you are a first-time user, you won't have any data trees to use, so you will have to create some. The command which creates OLPARS data trees, from user collected data, is called FILEIN. At some previous time, you will have had to create a system text file containing the data which you want to examine, within the format that is acceptable to FILEIN (see description of FILEIN*). If the system text file exists, invoke FILEIN to create your OLPARS data tree.

If you are interested in seeing which data trees are present in your OLPARS directory, use the LISTTREES command. It will give you an alphabetically ordered listing of the names of all the trees existing in your directory.

3.5 BEGINNING ANALYSIS - EXAMINING YOUR DATA

As a data analyst, you have apriori knowledge of the data you have collected (see section on importance of feature extraction). This "before-hand" knowledge can be augmented by obtaining some statistical information about the data from OLPARS. Therefore, one of the first operations to perform after converting your data into an OLPARS data tree is to print out your data using the print data set (PRTDS) utility.

* After "logging in" to OLPARS, anytime a command description is needed, it can be obtained by using the HELP command.

If you have not yet generated features to represent the objects to be classified, and are using the "raw measurements" or attributes taken directly from the environment, this command can show you each data vector, the range and overlap of each measurement in the data, the means and standard deviations of the collected classes, and give covariance and correlation matrices of each class in the data. The measurement ranges and overlap graphs (See Figure 3-1) are quite useful in determining whether or not there is any class distinction in the "raw measurements" themselves, before the data is examined in structure analysis, or put through the feature extraction process (Feature vectors can be analyzed with this method also).

3.6 TEST DATA SET A "MUST"

Every logic created should have its validity tested with data not used in the logic design.

If you have collected two sets of data, one for logic design and one for testing the logic design, then this next step can be omitted. This step consists of dividing the original data set into a design data set and test data set.

The command CRANDTS (create random test set) "randomly" selects vectors from the original data set and creates two new data sets; one for logic design and one to be used for testing that logic. When creating the test set, remember that enough vectors must remain in the design set to keep statistical validity intact.

TREE STRUCTURE OF DATA TREE grains

RANGES FOR grains (soy)

| NUMBER OF DIMENSIONS | 12 | MEASUREMENT | MINIMUM | MAXIMUM | RANGE |
|------------------------|----|-------------|----------|----------|---------|
| TOTAL NUMBER OF NODES | 9 | 1 | 156.0000 | 176.0000 | 10.0000 |
| | | 2 | 170.0000 | 181.0000 | 11.0000 |
| NUMBER OF LOWEST NODES | 7 | 3 | 191.0000 | 198.0000 | 7.0000 |
| | | 4 | 189.0000 | 196.0000 | 7.0000 |
| VECTOR | | 5 | 162.0000 | 178.0000 | 16.0000 |
| COUNT | | 6 | 161.0000 | 172.0000 | 11.0000 |
| | | 7 | 188.0000 | 194.0000 | 6.0000 |
| 293 grains | | 8 | 165.0000 | 177.0000 | 12.0000 |
| | | 9 | 178.0000 | 191.0000 | 13.0000 |
| | | 10 | 164.0000 | 179.0000 | 15.0000 |
| 42soy | | 11 | 154.0000 | 172.0000 | 18.0000 |
| 42corn | | 12 | 173.0000 | 187.0000 | 14.0000 |
| 45oats | | | | | |
| 42wheat | | | | | |
| 43Clov | | | | | |
| 37alfa | | | | | |
| 42rye | | | | | |

OVERLAP GRAPH FOR MEASUREMENT 12

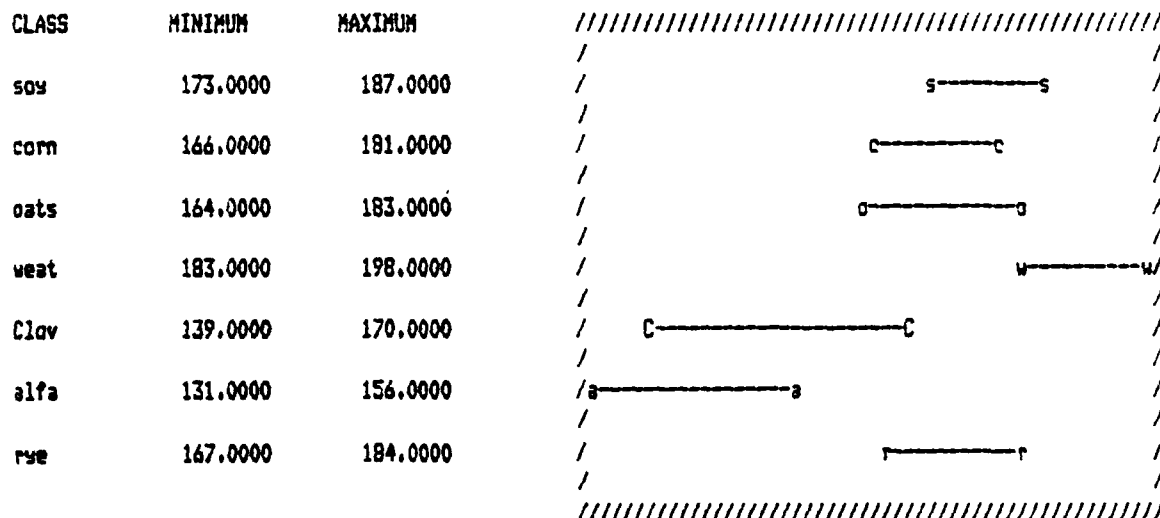


Figure 3-1 Some Available Information from PRTDS Command

3.7 STUDYING THE DATA STRUCTURE

At this point, you could choose to perform measurement evaluation or structure analysis on your design data set. This section deals with some ideas and examples of structure analysis.

The basic use of structure analysis in OLPARS is in determining the clustering properties of each data class; does the data class cluster around one point (unimodal data) or more than one point (multimodal data)? If the data is multimodal, it is frequently better to sub-divide the class into its component sub-classes before attempting to design logic for distinguishing between classes. This is particularly true if the logic to be designed is statistically based.

All the algorithms for structure analysis involve projecting the data onto a line (one-space projections) or a plane (two-space projection). If multimodality is present, the analyst is allowed to partition the projection space. (Note, these projections may also be used as the basis for group logic design).

For a first attempt at structure analysis, you might try one-space coordinate projections (S1CRDV). This projection will show you the spread of an individual measurement. The data class range information in this projection is identical to that of the data print overlap graph (for the same measurements), except with the coordinate projection display, the added histogram(s) shows you how the data clusters. Note, in the macro plot of Figure 3-2 which

USING OLPARS -- 3
STUDYING THE DATA STRUCTURE

depicts the GRAINS data set clustering of measurement twelve, that the "Clov" data class appears to have two separate clustering points about the class mean. The "intensified" micro plot (same figure), with the projected data class mean (below the base line) illustrates this clustering more effectively.

Another projection type to try is a two-space eigenvector projection. The eigenvectors and eigenvalues of the covariance matrix of the data set are generated. The two largest eigenvalues computed correspond to the two eigenvectors to be used for the projection space. The plane defined by these two eigenvectors minimizes the sum of the square of the distance of each vector from the plane (as done in least squares line fitting). This particular projection preserves the class clustering structure. It also preserves clustering that may occur within a class (a multimodal class). Thus, the eigenvector projection is a good tool to help locate multimodal data. Note, using this projection does not guarantee that all classes will be separable or that multiple mode classes can be found (other plane projections may show better separations for a particular class), but it is a good start at analyzing overall data structures.

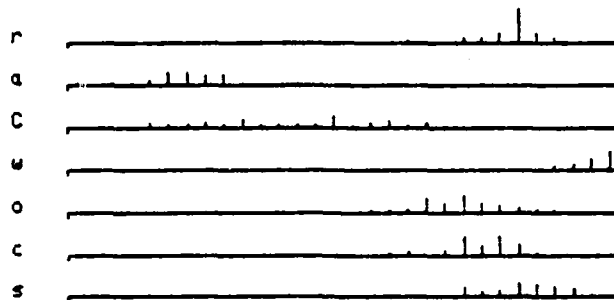
Figure 3-3 shows a two-space eigenvector projection for the GRAINS data set, along with a list of the computed eigenvalues for the data set covariance matrix. This projection has separated the classes "Clov", "alfa", "rye", and "weat" fairly well. The classes

* Mathematical evidence is shown in Appendix A.

MACRO PLOT (MEAS. 12) DATE: 20-JAN-82 11:11:39
 ZOOM SCALE
 VECTOR COUNTS

* S * C
 * O * U
 * C * G
 * P

BINWIDTH
 CDISPLAY
 CSCALE
 DBNDY
 DRAWBNDY
 INTENSIFY
 PROJMN
 RDISPLAY
 REDRAW
 RESTRUCT
 SCALZM
 SCALRET
 SELECT



MIN = 1.3100E+02 BIN SIZE = 2.000E+00
 MAX = 1.9100E+02 BINS = 30

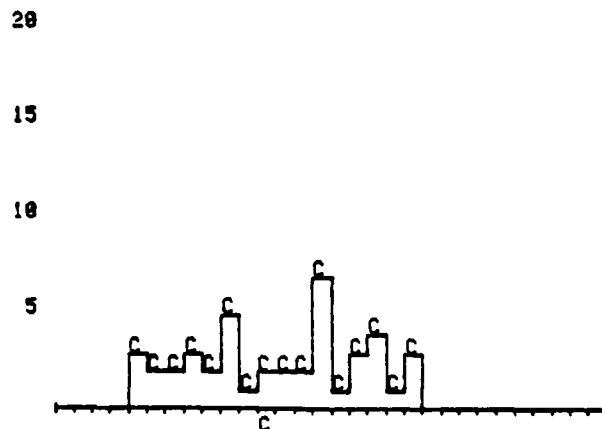
CURRENT
 OPTION:
 SICRDU

DATASET:
 grains

MICRO PLOT (MEAS. 12) DATE: 20-JAN-82 11:19:31
 ZOOM SCALE
 VECTOR COUNTS

* S * C
 * O * U
 * C * G
 * P

BINWIDTH
 CDISPLAY
 CSCALE
 DBNDY
 DRAWBNDY
 INTENSIFY
 PROJMN
 RDISPLAY
 REDRAW
 RESTRUCT
 SCALZM
 SCALRET
 SELECT



MIN = 1.3100E+02 BIN SIZE = 2.000E+00
 MAX = 1.9100E+02 BINS = 30

CURRENT
 OPTION:
 SICRDU

DATASET:
 grains

Figure 3-2 One-Space Coordinate Projections

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 9.884713E+02 |
| 2 | 2.755214E+02 |
| 3 | 7.694861E+01 |
| 4 | 7.991776E+00 |
| 5 | 4.302827E+00 |
| 6 | 2.421696E+00 |
| 7 | 1.747906E+00 |
| 8 | 1.626976E+00 |
| 9 | 1.401383E+00 |
| 10 | 1.345815E+00 |
| 11 | 8.582678E-01 |
| 12 | 7.110363E-01 |

PRINTOUT (Y/N)? N

EIGENVECTOR NO. FOR THE X PROJECTION: 1

EIGENVECTOR NO. FOR THE Y PROJECTION: 2

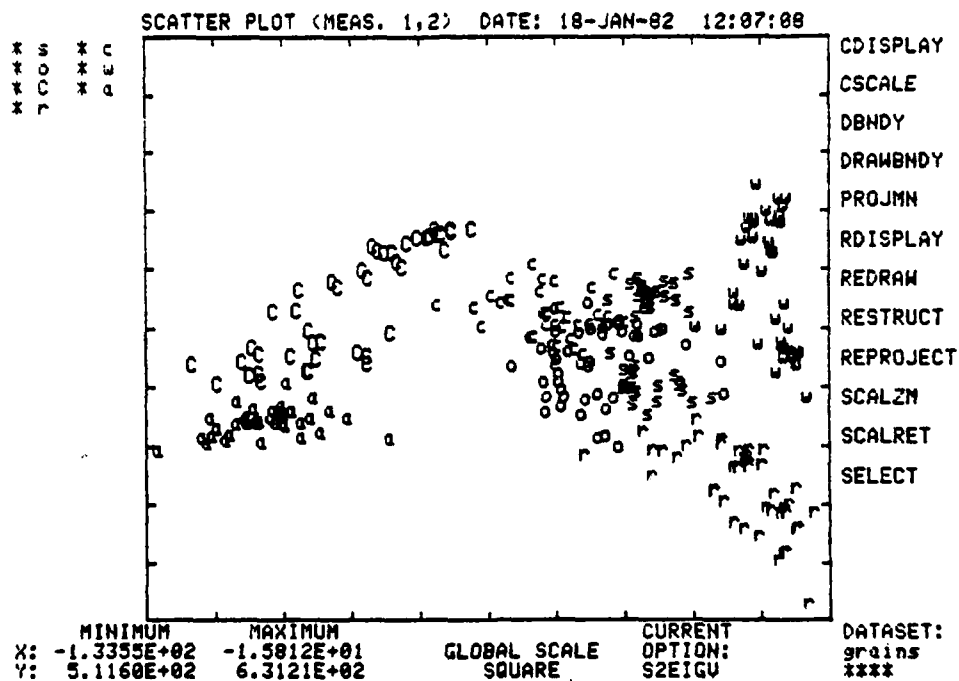


Figure 3-3 Eigenvalues and Two-Space Eigenvector Projection

"soy", "oats", and "corn" overlap. If you look at the next displays (Figure 3-4) which limits the number of classes shown at one time, you can see that the classes "weat", "soy" and "Clov" each cluster at two separate points, away from their respective means (class symbols located in rectangles). These classes are showing signs of bimodality. At this point, each class may be restructured into two subclasses reflecting the bimodal clustering.

3.3 RESTRUCTURING A DATA TREE WHY?

If multimodal classes have been found to exist in a data set, those classes should be divided into subclasses. This will aid in reducing the number of misclassified vectors that could appear during logic design. This process of creating data subclasses is termed "restructuring the data".

The restructuring process operates as follows. A multimodal class is chosen to be restructured. The projected vector display is sub-divided into regions* containing the subclasses. The regions are defined by partitions or boundaries drawn on the displays (see Figure 3-5 and DRAWBNDY) after subclass regions have been defined, the data class is re-structured by using the command RESTRUCT. Figure 3-6 shows a data tree display of the GRAINS data set after re-structuring each of the bimodal classes.

* The regions should be "convex" regions, as in the geometric term "convex" polygon.

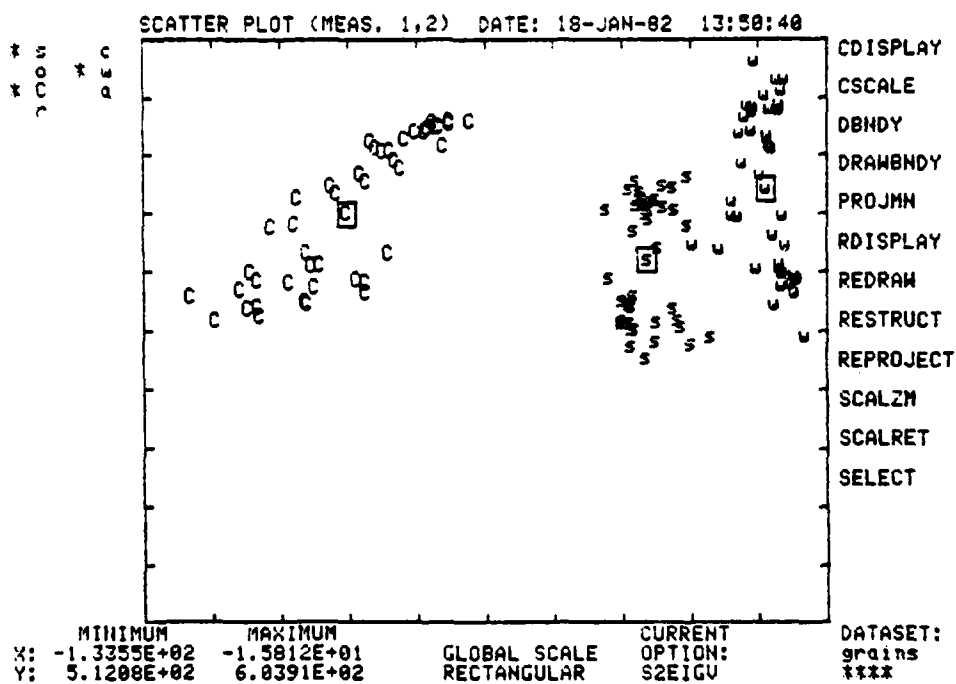
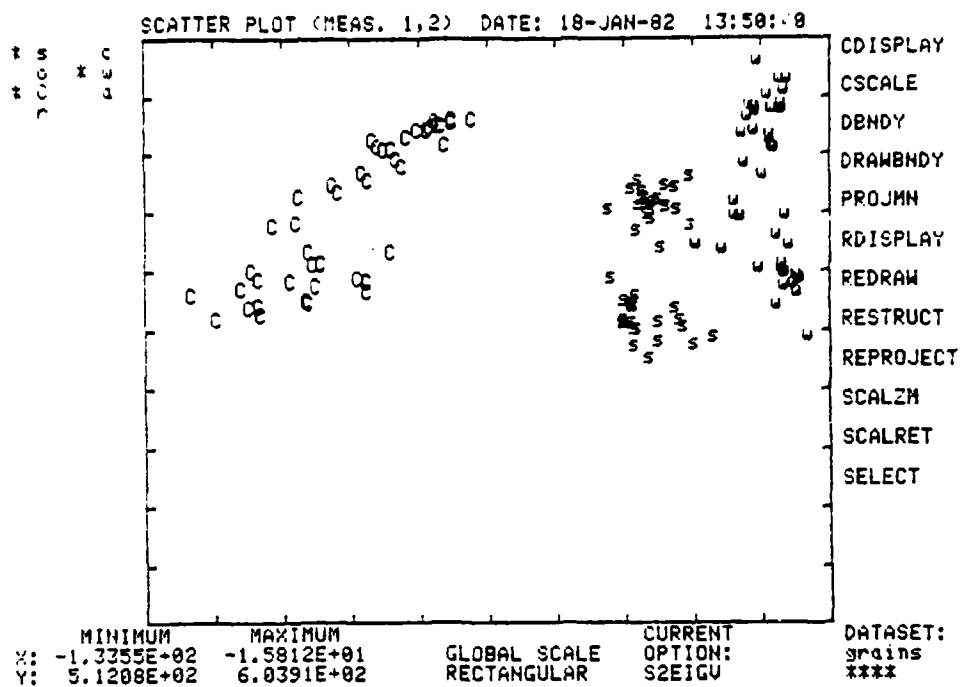


Figure 3-4 Two-Space Eigenvector Projection Scatter Plots
(with "selected" classes and projected class means)

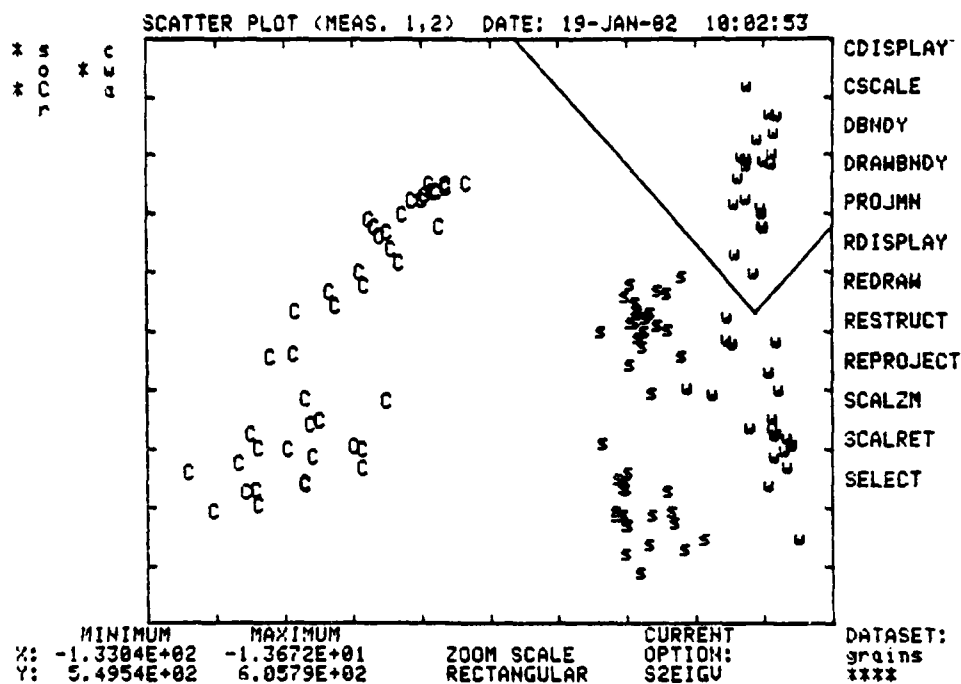
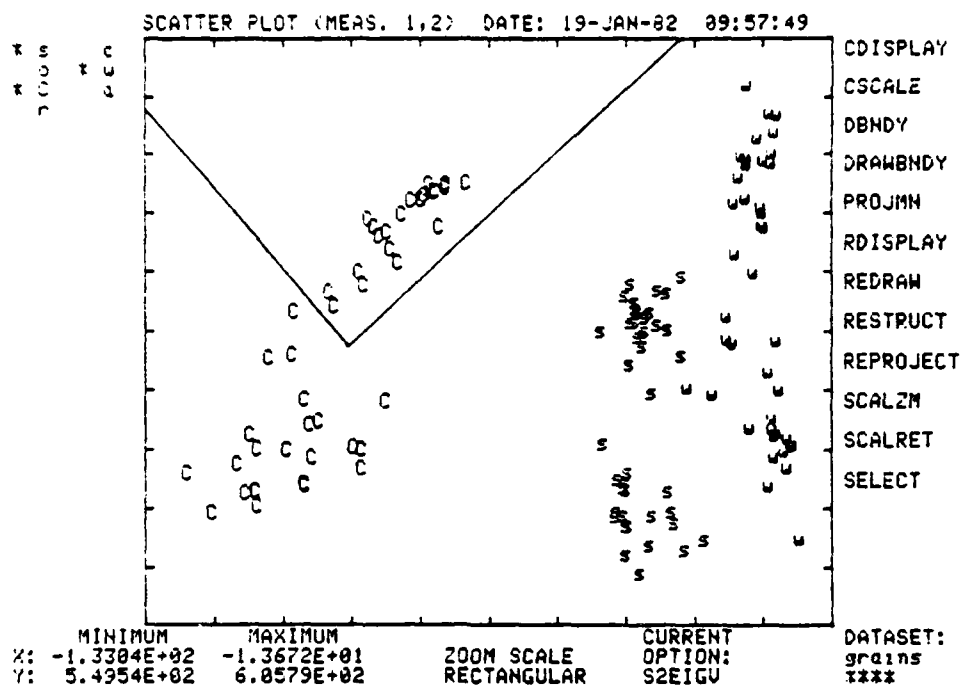


Figure 3-5 Data Partitionment on Two-Space Displays

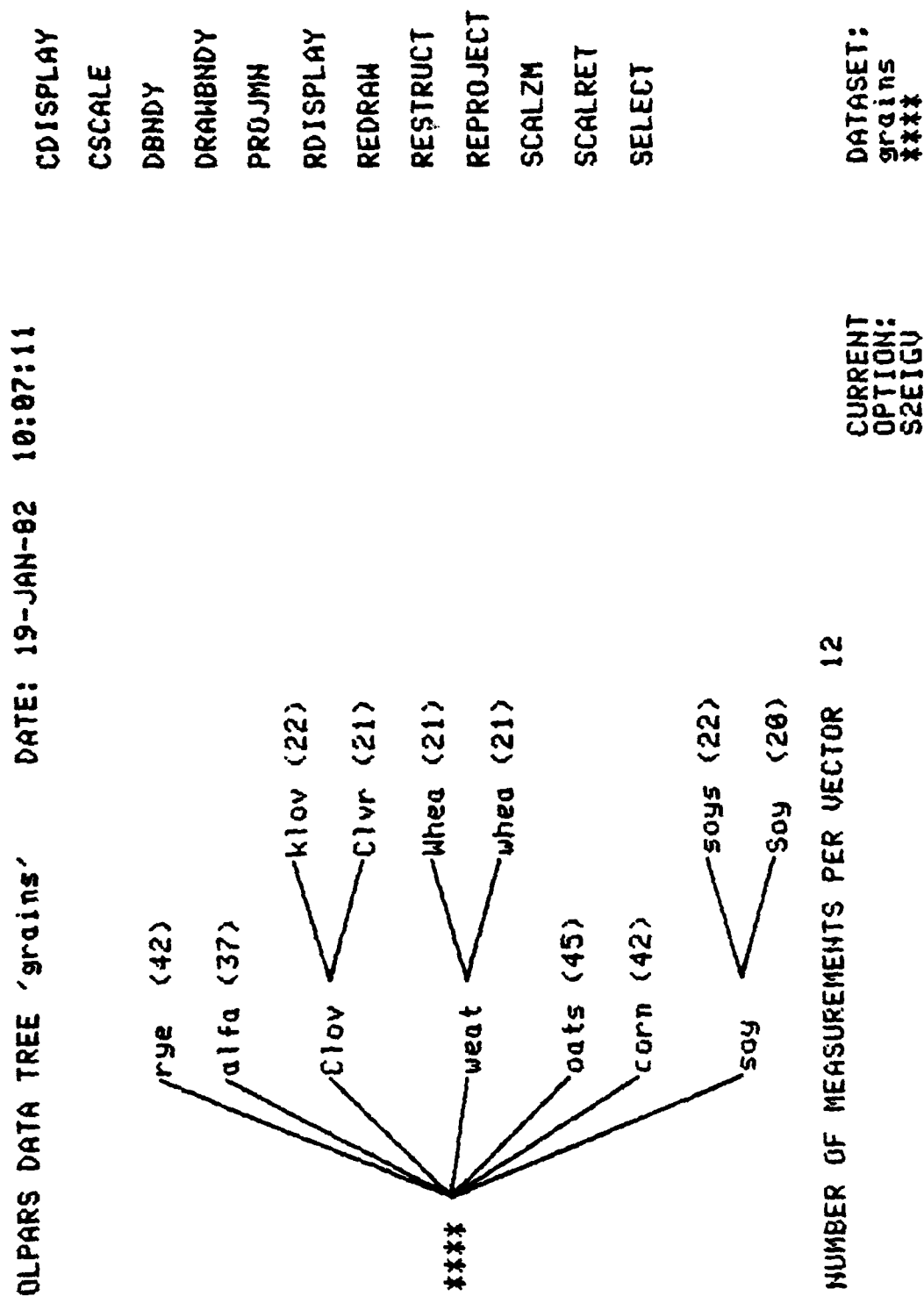


Figure 3-6 Restructured Data Tree

3.9 CHOOSING THE "BEST" MEASUREMENTS

It may be possible to reduce the number of features originally selected without affecting the clustering of the data. This is due to the fact that some features may not be useful in differentiating classes. Measurement evaluation algorithms aid in the selection of features which have the most discriminatory power. The discriminant measure algorithm (DSCRMEAS) is particularly useful for ranking a set of measurements when the class conditional probability distributions are approximately unimodal.

Measurement evaluation commands provide various types of rankings which enable an analyst to evaluate the measurements of a data set. Selection of the best measurements is a non-trivial problem that involves examining different types of rank order displays and combining this information with any apriori knowledge about the data. The "best" measurements may be selected, either by the analyst (SLCTMEAS) or through the use of the union by class and/or union by class pair algorithm(s) (UNION). The original data tree may then be transformed (TRANSFRM) to produce a new data tree containing only the selected features. This measurement reduction process can simplify and speed up analysis of data.

In Section 3.8, data classes showing bimodality were restructured into subclasses. Figure 3-7 shows that the restructured GRAINS data set appears to be approximately unimodal. Therefore, it might be appropriate at this time, to perform a

USING OLPARS -- 3
CHOOSING THE "BEST" MEASUREMENTS

discriminant evaluation on the data.

Figure 3-3 shows the output of the measurement evaluation. Measurements are ranked on the basis of their ability to discriminate all classes (overall discrimination). The largest discriminant values correspond to measurements with the most discriminatory power. The top display shows the results when the discriminant measurement value is calculated by weighting (multiplying) the variance of each measurement for a class by the number of vectors in the particular class. The bottom display shows the output when the variances are weighted equally. That is, the weighing factor is the total number of vectors in the data set is divided by the number of classes in the data set. The "equal weighting" of classes may be used when the results of the evaluation might be affected by a large difference in the number of vectors present in each class. For example, it may have been difficult to collect a large amount of data for a particular class. However, there is knowledge, or a reason to believe that the sample data available is a good representation of the population (real world). Consequently, the data should be weighted equally during measurement evaluation for better results.

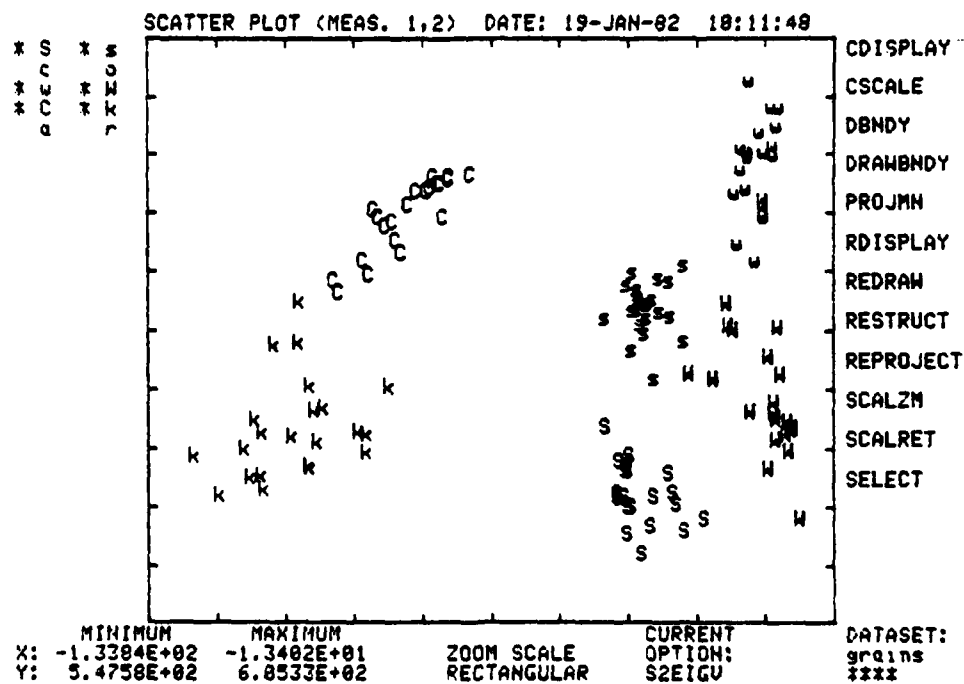
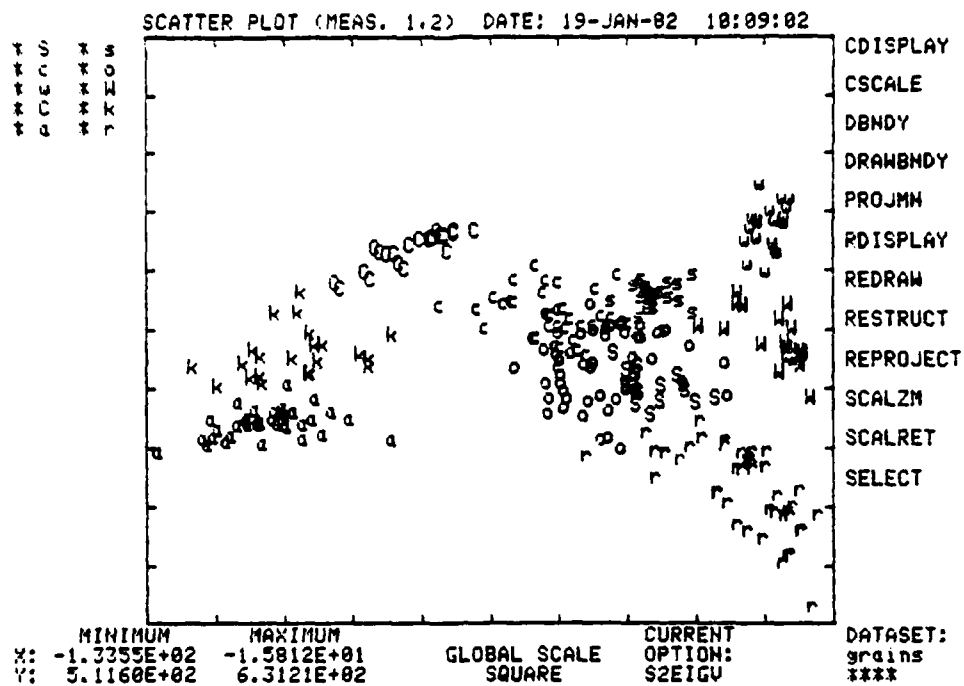


Figure 3-7 Two-Space Data Projection after Restructuring

A RANK ORDER DISPLAY

DATE: 28-JAN-82 11:38:20

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| 9 | 7.8259E+01 | C | W/C |
| 8 | 7.1279E+01 | C | W/C |
| 12 | 6.8153E+01 | k | W/k |
| 11 | 6.5765E+01 | W | W/a |
| 6 | 5.3037E+01 | C | S/C |
| 5 | 3.6815E+01 | S | S/C |
| 1 | 3.6739E+01 | S | S/W |
| 10 | 3.2301E+01 | W | W/C |
| 7 | 3.0449E+01 | C | S/C |
| 2 | 2.8300E+01 | S | S/C |
| 4 | 2.8213E+01 | C | S/C |
| 3 | 1.6947E+01 | C | S/C |

RANK
SLCTMEAS
TRANSFRM
UNION

CURRENT
OPTION:
DSCRMEAS

DATASET:
GRAINS

A RANK ORDER DISPLAY

DATE: 28-JAN-82 11:39:18

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| 9 | 6.5931E+01 | C | W/C |
| 12 | 6.2497E+01 | a | W/a |
| 11 | 6.1464E+01 | a | W/a |
| 8 | 6.1387E+01 | C | W/C |
| 6 | 4.7576E+01 | C | S/C |
| 5 | 3.2553E+01 | C | S/C |
| 1 | 3.0567E+01 | S | S/W |
| 10 | 2.9497E+01 | C | W/C |
| 7 | 2.8588E+01 | C | S/C |
| 4 | 2.5014E+01 | C | S/C |
| 2 | 2.3873E+01 | C | S/C |
| 3 | 1.5561E+01 | C | C/r |

RANK
SLCTMEAS
TRANSFRM
UNION

CURRENT
OPTION:
DSCRMEAS

DATASET:
GRAINS

Figure 3-8 Overall Rank Order Displays
(top showing vector count weighting)
(bottom showing equalized weighting)

For the GRAINS data set, there is no significant difference between weighting class variances equally or weighting by the number of vectors per class*, because there is not a large vector difference between the class samples gathered.

In the remainder of rank order example figures, the discriminant measurement values are calculated by weighting the class variances equally.

3.9.1 Selection Of Measurement By The Analyst -

After obtaining some class discriminating measurements (see previous section) a data transformation using a portion of the overall best measurements could be performed to carry the analysis further, other types of rank order displays can be obtained (see RANK command description).

Figure 3-7 (previous section) shows that class "oats" overlaps with several classes; predominantly with classes "soy" and "corn". A measurement ranking of class "oats" orders the measurements on the basis of their ability to distinguish class "oats" from all other classes. Figure 3-9 shows that measurement 8 is the best measurement for distinguishing class "oats" from all other classes. Since measurement 8 is also one of the overall best measurements

* In Figure 3-8, the two rank order displays show that measurements 9, 8, 12, and 11 remain the top four measurements, and that the other measurements (except measurement 2) remained in their relative order.

USING CLPARS -- 3 CHOOSING THE "BEST" MEASUREMENTS

(see Figure 3-8), it could be a candidate measurement for a data transformation.

The scatter plots in Figure 3-10 show the overlapping of class "oats" with classes "soy" and "corn". Figure 3-11 shows two rank order displays. The first display is a measurement ranking of the class pair "oats" and "soy"; the second is a measurement ranking of the class pair "oats" and "corn".

These rankings order the measurements on the basis of their ability to separate class "oats" from class "soy" and class "oats" from class "corn", respectively. Note that measurements 4 and 1 are the best measurements for separating class "oats" from class "soy", and measurement 10 is the best measurement for separating class "oats" from class "corn". Although these measurements were not ranked very favorably in the overall evaluation, they can be included in a data transformation because of their ability to separate these classes which overlap.

3.9.1.1 "Automatic" Selection Of Measurements -

A large number of data classes and/or a large vector dimensionality may make it difficult for the analyst to select measurements. If, after analyzing your data and looking at rank order displays, you cannot determine which measurements to choose. Two options are available to automatically select measurements: (1) Union by class and (2) Union by class pair (both found in the UNION command). The "union by class" algorithm performs a ranking

```

A RANK ORDER DISPLAY      DATE: 19-JAN-82  15:56:08
      A MEASUREMENT RANKING OF CLASS 0
      MEAS      VALUE
      8      2.2488E+00
      12     2.0881E+00
      11     1.7342E+00
      9      1.7072E+00
      10     1.3248E+00
      4      9.5549E-01
      5      7.0785E-01
      7      6.9444E-01
      2      4.9875E-01
      1      4.5922E-01
      6      4.5759E-01
      3      3.7473E-01

RANK      SLCTMEAS      TRANSFRM      UNION
CURRENT   OPTION:
DSCRMEAS
DATASET:
GRAINS
***

```

Figure 3-9 Rank Order Display for Single Class

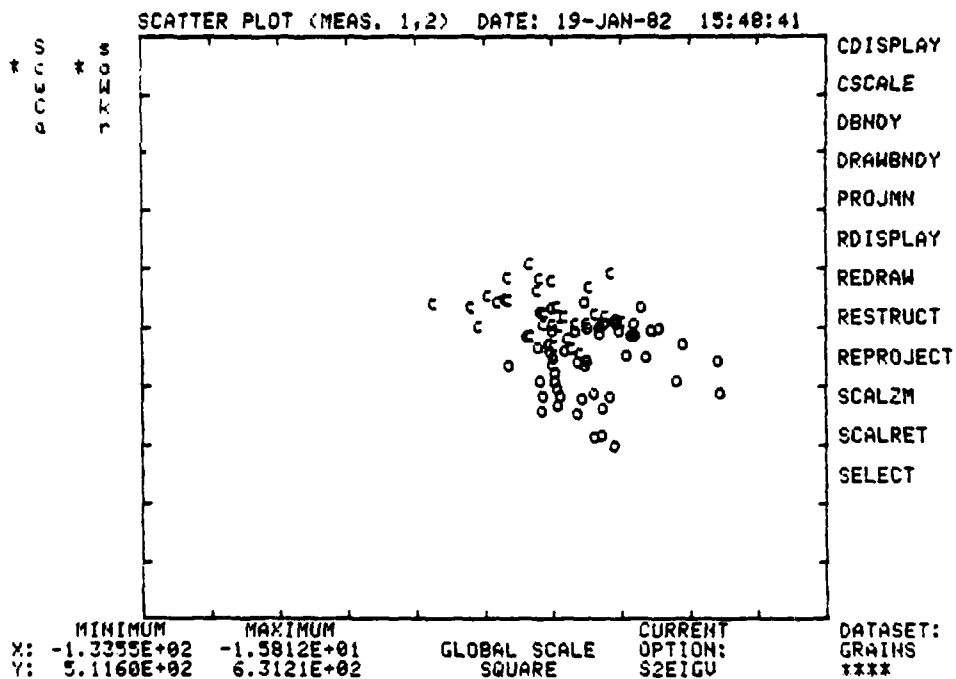
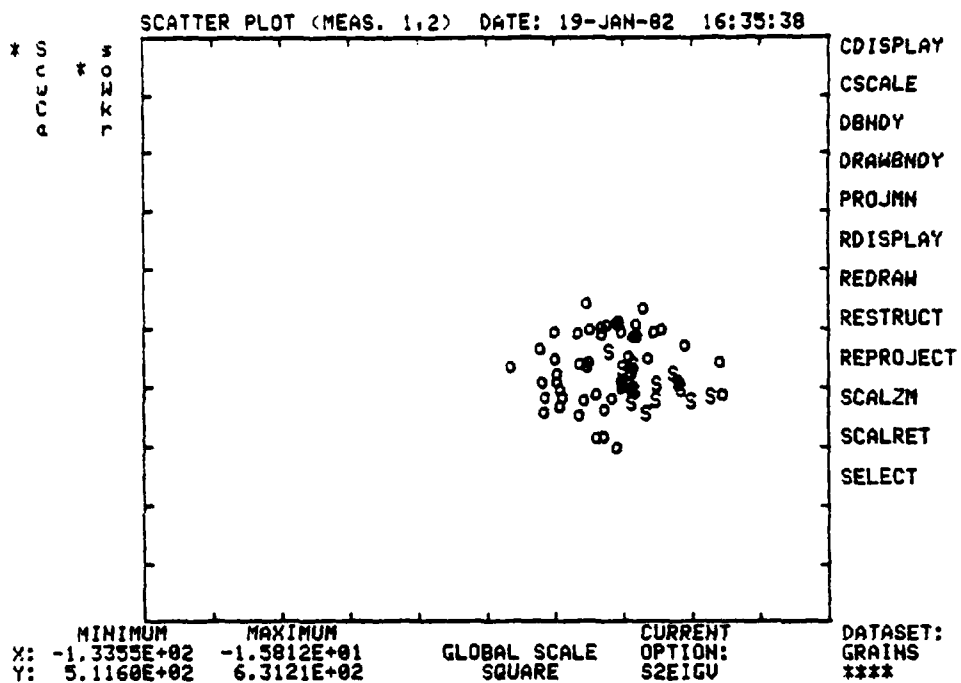


Figure 3-10 Scatter Plots Showing Class Overlays

A RANK ORDER DISPLAY DATE: 20-JAN-82 09:39:19

A MEASUREMENT RANKING OF CLASS PAIR o/s

| MEAS | VALUE |
|------|------------|
| 4 | 1.8754E-01 |
| 1 | 1.8152E-01 |
| 5 | 1.6648E-01 |
| 2 | 1.6461E-01 |
| 3 | 7.6756E-02 |
| 6 | 7.5643E-02 |
| 10 | 2.6829E-02 |
| 7 | 2.2841E-02 |
| 12 | 2.0734E-02 |
| 11 | 1.6893E-02 |
| 9 | 9.4119E-04 |
| 8 | 3.0382E-05 |

RANK
SLCTMEAS
TRANSFRM
UNION

CURRENT
OPTION:
DSCRMEAS

DATASET:
GRAINS

A RANK ORDER DISPLAY DATE: 19-JAN-82 16:02:24

A MEASUREMENT RANKING OF CLASS PAIR o/c

| MEAS | VALUE |
|------|------------|
| 10 | 2.7456E-01 |
| 8 | 1.3466E-01 |
| 9 | 1.2429E-01 |
| 1 | 6.4682E-02 |
| 2 | 1.6827E-02 |
| 3 | 8.0116E-03 |
| 7 | 4.2748E-03 |
| 6 | 4.1665E-03 |
| 12 | 3.2931E-03 |
| 5 | 2.0438E-03 |
| 4 | 1.5792E-03 |
| 11 | 1.2915E-03 |

RANK
SLCTMEAS
TRANSFRM
UNION

CURRENT
OPTION:
DSCRMEAS

DATASET:
GRAINS

Figure 3-11 Rank Order Displays for Class Pairs

USING OLPARS -- 3
CHOOSING THE "BEST" MEASUREMENTS

of measurements for each class and selects the measurement that was the best (largest) from each ranking. The result is the union (as in set notation) of the best measurements for distinguishing single classes from all other classes. The "union by class pair" algorithm is identical to the "union by class algorithm", except that measurements are ranked for each class pair, rather than for each class. The result is the union of the best measurements which distinguish one class from another class.

Figure 3-12 shows two more rank order displays. The top is the result of a union by class; the bottom is the result of a union by class pair. The asterisks on the display indicate which measurements were selected by the algorithms.

3.9.1.2 Measurement Reduction Transformation -
(Following Measurement Evaluation)

The measurement evaluation procedure will most likely consist of a combination of the two previously described techniques, manual selection of measurements by the analyst, and "automatic" selection of measurements by commands. For instance, measurement evaluation could begin with an examination of various rank order displays. The analyst studies the displays (rankings) in an effort to determine which measurements best separate classes. The second phase of evaluation could be a selection of measurements using the union by class and/or union by class pair function(s) ("automatic"

A RANK ORDER DISPLAY

DATE: 19-JAN-82 13:41:49

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| * 9 | 7.8259E+01 | C | W/C |
| * 9 | 7.1279E+01 | C | W/C |
| * 12 | 6.8153E+01 | K | W/K |
| * 11 | 6.5765E+01 | W | W/d |
| * 6 | 5.3837E+01 | C | S/C |
| 5 | 3.6815E+01 | S | S/C |
| 1 | 3.6739E+01 | S | S/W |
| 10 | 3.2301E+01 | W | W/C |
| 7 | 3.0449E+01 | C | S/C |
| 2 | 2.8300E+01 | S | S/C |
| 4 | 2.8213E+01 | C | S/C |
| 3 | 1.6947E+01 | C | S/C |

RANK
SLCTMEAS
TRANSFRM
UNION

CURRENT
OPTION:
OSCRMEAS

DATASET:
grains

A RANK ORDER DISPLAY

DATE: 19-JAN-82 13:42:38

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| * 9 | 7.8259E+01 | C | W/C |
| * 9 | 7.1279E+01 | C | W/C |
| * 12 | 6.8153E+01 | K | W/K |
| * 11 | 6.5765E+01 | W | W/d |
| * 6 | 5.3837E+01 | C | S/C |
| 5 | 3.6815E+01 | S | S/C |
| 1 | 3.6739E+01 | S | S/W |
| 10 | 3.2301E+01 | W | W/C |
| 7 | 3.0449E+01 | C | S/C |
| 2 | 2.8300E+01 | S | S/C |
| 4 | 2.8213E+01 | C | S/C |
| 3 | 1.6947E+01 | C | S/C |

RANK
SLCTMEAS
TRANSFRM
UNION

CURRENT
OPTION:
OSCRMEAS

DATASET:
grains

Figure 3-12 Rank Order Displays Showing Measurement Selection

USING OLPARS -- 3
CHOOSING THE "BEST" MEASUREMENTS

selection). The results of the second phase of evaluation will either verify the analyst's initial findings or provide additional measurements for selection. A final decision as to which measurements will be chosen may be based on the analyst's observations in combination with the results of the union functions. Following measurement evaluation, those measurements that were selected will be used to create a new data tree.

An example of a measurement selection procedure, followed by a measurement reduction transformation is described in the following text. For the GRAINS data set, measurements 1, 4, and 10 were selected as a result of the analysis previously done on class "oats", and because they were selected by the union by class pair algorithm. Measurements 8, 9, 11, and 12 were chosen because they are the "best overall" measurements, and because they were selected by both the union by class and union by class pair algorithms. For this example, measurements were selected using the command SLCTMEAS (see Figure 3-13). After measurements were selected, a data transformation was performed by the command TRANSFRM. The top of Figure 3-14 shows the new data tree called NGRAINS (new GRAINS). The structure of the new tree is identical to that of GRAINS. The number of measurements per vector, however, is only seven. These seven measurements correspond to measurements 1, 4, 8, 9, 10, 11 and 12 in the GRAINS data tree.

```

A RANK ORDER DISPLAY      DATE: 03-FEB-82  10:09:15

      AN OVERALL RANKING

MEAS  VALUE      CLASS  CLASS PAIR
* 9    6.5931E+01    C      W/C
* 12   6.2497E+01    a      w/a
* 11   6.1464E+01    a      w/a
* 8    6.1387E+01    C      W/C
6    4.7576E+01    C      S/C
5    3.2553E+01    C      S/C
1    3.0567E+01    S      S/W
10   2.9497E+01    C      W/C
7    2.8588E+01    C      S/C
4    2.5014E+01    C      S/C
2    2.3873E+01    C      S/C
3    1.5561E+01    C      C/r

RANK
SLCTMEAS
TRANSFRM
UNION

```

```

CURRENT
OPTION:
DSCRMEAS

DATASET:
GRAINS
***

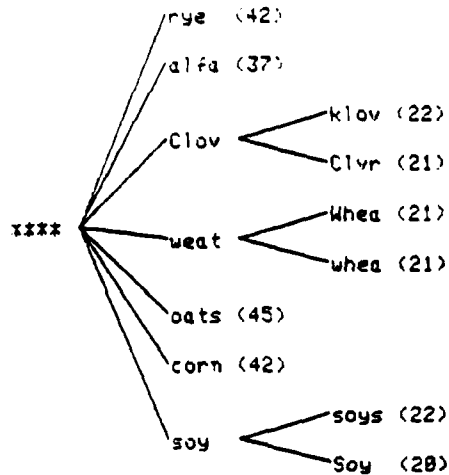
```

Figure 3-13 Rank Order Display of Measurements to be used in a Data Transformation

OLPARS DATA TREE (ngrains)

DATE: 29-JAN-82 16:20:00

ANYTHING



NUMBER OF MEASUREMENTS PER VECTOR 7

CURRENT
OPTION:
ANYTHING

DATASET:
ngrains

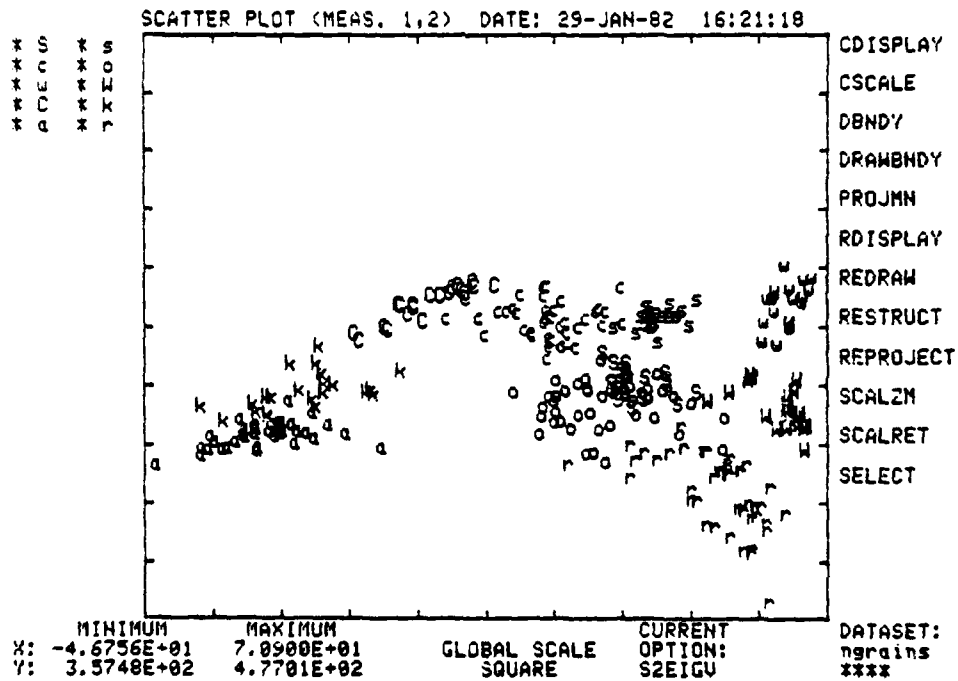


Figure 3-14 Data Tree Display and Scatter Plot
Projection After Measurement Reduction

As mentioned previously, the purpose of measurement evaluation and measurement reduction is to eliminate non-discriminating or non-essential measurements from a data set. If the most discriminating measurements are present in the new data tree, the clustering of the data should not be affected. On the other hand, if after a data transformation, clustering is no longer present, then important measurements have been eliminated. The bottom of Figure 3-14 shows an eigenvector projection of the data set NGRAINS. It can be seen that the elimination of measurements 2, 3, 5, 6, and 7 did not greatly change the clustering of the data (compare with Figure 3-7). Consequently, the newly created data set NGRAINS can be used for logic designing.

3.9.1.3 A Final Note About Measurement Evaluation -

Evaluating measurements for the purpose of reducing the data set dimensionality can be an iterative process. In the previous example, measurements were selected primarily on the basis of their "overall goodness," that is, they demonstrated an ability to separate classes. If one or more classes cannot be separated after designing and evaluating logic, it will be necessary to re-evaluate the measurements in the original data set in an attempt to find discriminating measurements for these classes.

USING OLPARS -- 3
CHOOSING THE "BEST" MEASUREMENTS

3.9.2 Data Transformations -

In addition to a measurement reduction transformation (performed in conjunction with measurement evaluation computations), a data set within OLPARS may be transformed by any of the following three independent transformations: normalization, eigenvector, or measurement transformation. Upon execution of any of these algorithms, a new tree containing the transformed vectors is created. The new tree will have the same structure as the original tree.

3.9.2.1 The Normalization Transformation -

The normalization transformation determines the standard deviation along each coordinate measurement of the selected data set. Each vector component within the data set is then modified by dividing it by this corresponding standard deviation. The resulting normalized data set will have unit variance along each coordinate measurement. It may be necessary to normalize a data set when extremely large or small data values cause the results of numerical calculations to be inaccurate (i.e., numerical "round off" problems).

3.9.2.2 Eigenvector Transformation (Data Reduction) -

The eigenvector transformation computes the eigenvalues and eigenvectors of the covariance matrix of the selected data set. Those eigenvectors corresponding to eigenvalues with values greater

than a user-specified threshold are used to transform the data set, that is, each vector from the data set is projected onto each of the selected eigenvectors. The resulting scalar number(s) obtained from the projection become measurements in a new vector. This vector represents a mapping of the original data vector onto the eigenvector subspace defined by the user. The eigenvector subspace provides a least squares fit to the selected data set, since the sum of the squared residual distances from the subspace is minimized. The error in fitting the data can be determined by summing the remaining (unselected) eigenvalues. The transformation essentially involves an orthonormal rotation of the basis vectors of the data set until they are aligned with the eigenvectors.

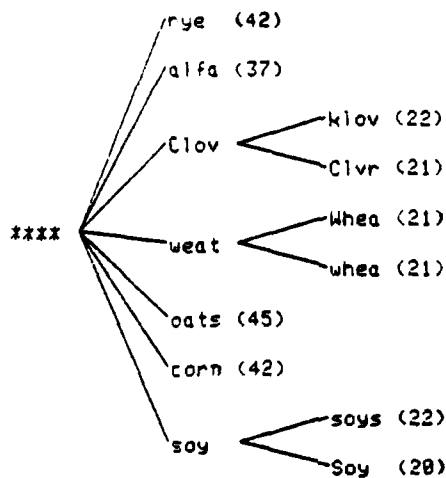
This technique has proven useful both as a research tool and as an aid to structure analysis and logic design. Measurement reduction may also be performed through the use of the eigenvector transformation. The newly created data set will have a dimensionality equal to the number of eigenvectors used in the transformation. An example of a data reduction using the eigenvector transformation is described in the following text.

The top of Figure 3-15 lists the eigenvalues for the data set, GRAINS. The user has typed in a threshold of eigenvalue number of 6 which will cause the eigenvectors corresponding to the six largest eigenvalues to be used in the transformation. The bottom of Figure 3-15 shows the data tree EGRAINS which was created as a result of the transformation. The dimensionality of EGRAINS is 5. The display in Figure 3-16 shows a coordinate projection of

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 9.884713E-02 |
| 2 | 2.755214E+02 |
| 3 | 7.694861E+01 |
| 4 | 7.991776E+00 |
| 5 | 4.382827E+00 |
| 6 | 2.421696E+00 |
| 7 | 1.747906E+00 |
| 8 | 1.626976E+00 |
| 9 | 1.401383E+00 |
| 10 | 1.345815E+00 |
| 11 | 8.502678E-01 |
| 12 | 7.110363E-01 |

PRINTOUT (Y/N)? N
NUMBER (POSITION) OF THE THRESHOLD EIGENVALUE: 6

OLPARS DATA TREE 'EGRAINS' DATE: 05-FEB-02 14:19:14 ANYTHING



NUMBER OF MEASUREMENTS PER VECTOR 6

CURRENT
OPTION:
ANYTHING

DATASET:
EGRAINS

Figure 3-15 Eigenvalues and Data True After
an Eigenvector Transformation

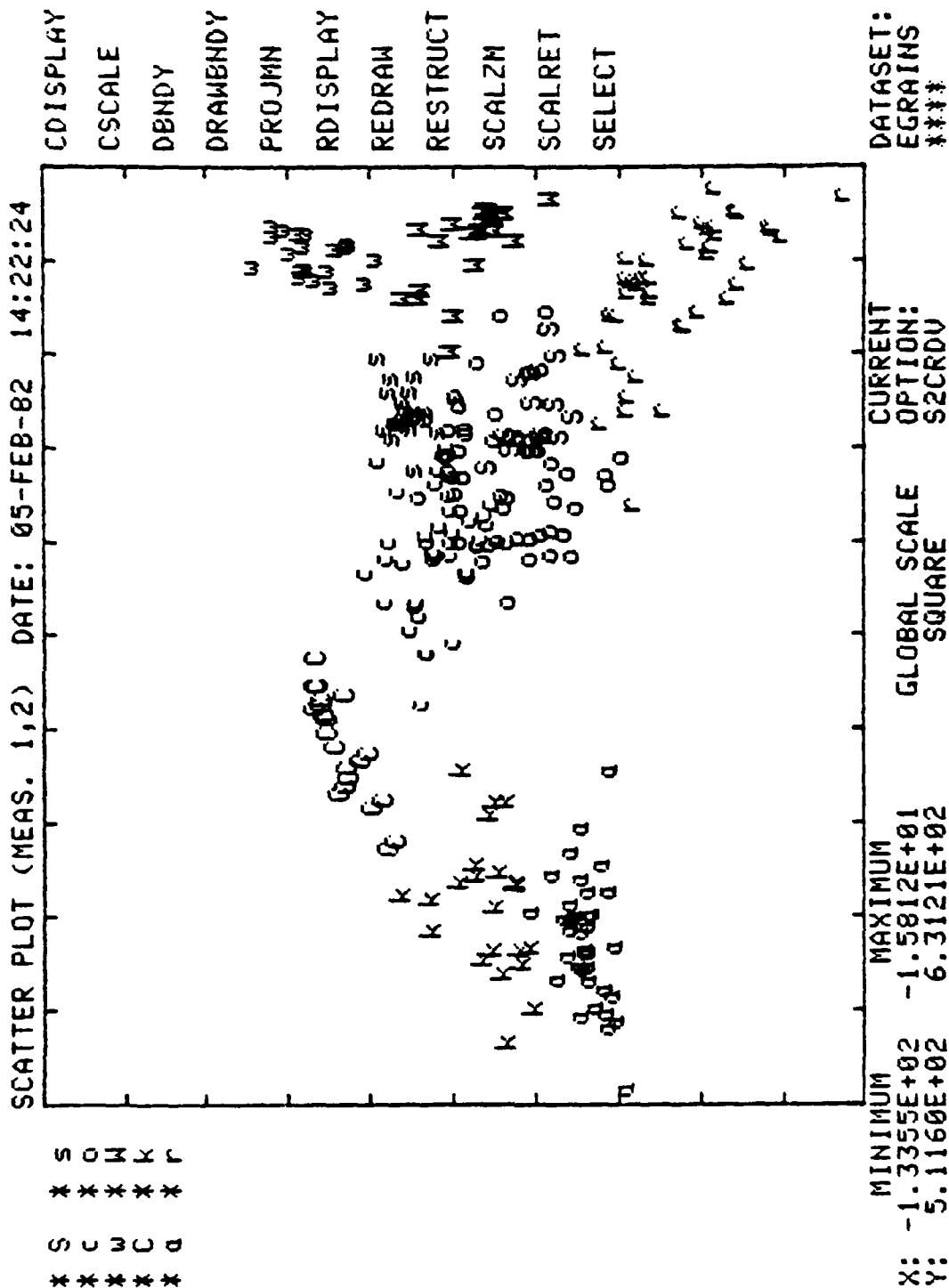


Figure 3-16 Coordinate Projection Plot of a Data Tree
After an Eigenvector Transformation

USING CLPARS -- 3
CHOOSING THE "BEST" MEASUREMENTS

measurements 1 and 2 from the new data tree EGRAINS. Notice that this display is identical to the eigenvector projection plot in which the data set GRAINS was projected onto the two largest eigenvectors (see Figure 3-7)

3.9.2.3 Measurement Transformation -
(Data Combination Or Alteration)

A measurement transformation allows the user to define new features which are functions of the original features. The new features are defined using FORTRAN statements which are placed into a FORTRAN subroutine by the user (see the MEASXFRM Command Description). The measurement transformation option provides the OLPARS user with practically an unlimited capability for defining both linear and nonlinear transformations. Once the new features have been defined, the transformation is performed, and a new tree containing vectors composed of the new user-defined features is created.

A measurement transformation may be performed to rescale one or more measurements in a data set. The scaling factor may be a function defined by the user, such as a square root or logarithmic function, or the value of one or more measurements, or a combination of both of these. A measurement transformation can also be used to combine linearly dependent measurements into single measurements, or to eliminate unwanted measurements from a data set.

3.10 LOGIC DESIGN AND EVALUATION

The OLPARS Logic Design facilities provide extensive mathematical/graphical techniques for allowing the user to tailor decision logic to the structure of the class data. In general, pattern classification is undertaken following a pattern analysis conducted on each of the data classes for which logic is to be designed. The purpose of this analysis is to ensure that each data class is unimodal; that is, the vectors from each class are clustered in one region of the measurement space. Although not always required, the unimodality property is highly desirable in order to ensure an effective logic design. In those cases where the class data is found to be multimodal, our philosophy dictates that each mode be identified and the sample vectors corresponding to each mode be grouped as a named subclass (see Section 3.8). Upon completion of the logic design, the decision region in the measurement space corresponding to each subclass can be reidentified with the original multimodal classes.

It should be noted at this time that there is no straightforward well-defined procedure for designing logic. Several combinations of algorithms may be selected to produce the desired results. For the most part, decisions about which logic design algorithm to use will be based on the philosophy that between-group logic should be used to separate non-overlapping classes, and complete within-group logic should be used to separate statistically overlapping classes. Within these two categories of logic, decisions about which algorithm to choose may seem

arbitrary, and a final decision may result through a trial and error evaluation process. Figure 3-17 shows choices of logic design algorithms, based on the number of classes and number of vectors in the data set. The table in the figure can serve as a guideline for choosing a logic design algorithm.

3.10.1 Designing Logic -

In Section 3.8 the data set GRAINS was restructured to make each data class unimodal. In Section 3.9.1.2, a measurement reduction transformation eliminated non-essential measurements from the data set. The reduced data set was called NGRAINS. This section describes a procedure for designing logic using the modified data set NGRAINS and evaluating the logic using a test data set called, GRAINTST.

The first step* in designing logic is to create a logic tree using the command NAMELOG. The initial logic tree will consist of only one node with all the classes in the design data set present at this node. Figure 3-18 shows the logic tree GRAINLOG which was created using the data set NGRAINS. Note that all ten classes from the data set NGRAINS are present at logic node 1 of GRAINLOG. The logic design process will be completed when each logic node has only one class present.

* This assumes that the current design data set has been changed to NGRAINS.

"Rule Of Thumb"

Choices Of Logic Design

| | * LARGE DATA SAMPLE | * SMALL DATA SAMPLE |
|---------------------------|-------------------------------|---------------------------|
| * SMALL CLASS COUNT | FISHER | NRSTNBR |
| * LARGE CLASS COUNT | GROUP LOGIC THEN FISHER | NMV NRSTNBR |

* Small and large are subjective terms and are related to the type of data and vector dimensionality. Therefore, the analyst should decide whether the class count and data sample is small or large.

Figure 3-17 Logic Design Choices

OLPARS LOGIC TREE 'grainlog'
 CLASSES PRESENT: SscowMCKar
 DATE: 26-JAN-82 11:52:29
 ANYTHING

| | |
|------------|---------|
| 1 | INCMPLT |
| SscowMCKar | |

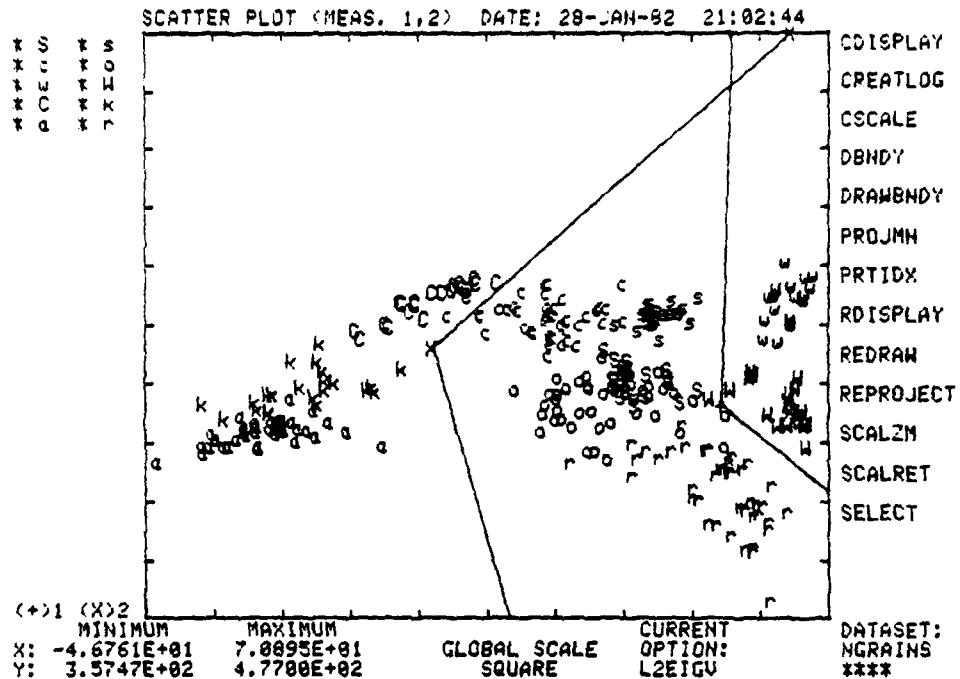
DESIGN DATA SET: ngrains (****)
 CLASS COUNT: 10
 DIMENSIONALITY: 7
 CURRENT
 OPTION:
 DLSUBSTR
 DATASET:
 ngrains

Figure 3-18 Logic Tree Formed via NAMELOG

3.10.2 "Between-Group Logic" -

Previous structure analysis on the data set NGRAINS has shown that some classes do not overlap. Therefore, to adhere to the basic philosophy of logic design, the next step entails using a between-group logic design algorithm to separate the non-overlapping classes. Figure 3-19 shows a scatter plot of an eigenvector projection produced by the command L2EIGV. The display also shows user-drawn boundaries which partition the display into 3 regions.

After boundaries have been drawn, the command CREATLOG is used to create the logic and evaluate the logic using the design data set vectors residing at the logic node (called a partial evaluation). The confusion matrix display in Figure 3-19 shows the results of the partial evaluation. The user has specified that the rightmost region will contain classes "Whea" and "whea"; the middle regions will contain classes "corn", "oats", "rye", "Soy", and "soys"; and the leftmost region will contain classes "alfa", "klov", and "Clvr". As would be expected, two vectors were misclassified. These vectors can be identified on the scatter plot in Figure 3-19. The vector from class "corn" in the leftmost region of the display, and the vector from class "whea" in the middle region of the display are the misclassified vectors. The overall results show that out of 293 vectors, 291 (or 99.3 percent) were correctly classified. For our purposes the number of correctly classified vectors is sufficient to accept the logic.



CONFUSION MATRIX (BTWN. GROUP LOGIC) DATE: 28-JAN-82 21:04:11

| REGION | LOGIC | DISPLAY SYMBOLS OF | LOGIC NAME- GRAINLOG | CDISPLAY |
|-------------|----------------|--------------------|--|-----------|
| | NODE | ASSOCIATED CLASSES | | CREATLOG |
| CONVEX (1) | 4 | Ww | | CSCALE |
| CONVEX (2) | 3 | corSs | | DBNDY |
| EXCESS | 2 | akC | | DRAWBNDY |
| CLASS NAMES | LOGIC (PARENT) | NODES (CHILDREN) | SUMS AND PERCENTAGES (CORRECT) (ERROR) | PROJMH |
| | 1 | 4 | COUNT PRCT | PRTIDX |
| | | 3 | COUNT PRCT | RDISPLAY |
| | | 2 | COUNT PRCT | REDRAW |
| soy | 20 | 0 | 20 100.0 | REPROJECT |
| soys | 22 | 0 | 22 100.0 | SCALZH |
| corn | 42 | 0 | 41 97.6 | SCALRET |
| oats | 45 | 0 | 45 100.0 | SELECT |
| whea | 21 | 21 | 21 100.0 | |
| whea | 21 | 20 | 20 95.2 | |
| clvr | 21 | 0 | 21 100.0 | |
| klov | 22 | 0 | 22 100.0 | |
| alfa | 37 | 0 | 37 100.0 | |
| rye | 42 | 0 | 42 100.0 | |
| TOTAL | 293 | 41 | 291 99.3 | |
| CORRECT | | 41 | | |
| (PRCT) | | 100.0 | | |
| ERRORS | | 0 | | |
| (PRCT) | | 0.0 | | |

LISTING OF MISCLASSIFIED VECTORS (Y/N)? n
RESULTS OK (Y/N)? y

CURRENT
OPTION:
L2EIGU

DATASET:
NGRAINS

Figure 3-19 Creating Between-Group Logic on 2-Space Display

AD-A118 733

PAR TECHNOLOGY CORP. NEW HARTFORD NY

F/G 9/2

ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. USE--ETC(U)

JUN 82 S E HAEHN; D MORRIS

UNCLASSIFIED

PAR-82-21

NL

2 of 4

AD-A
118753

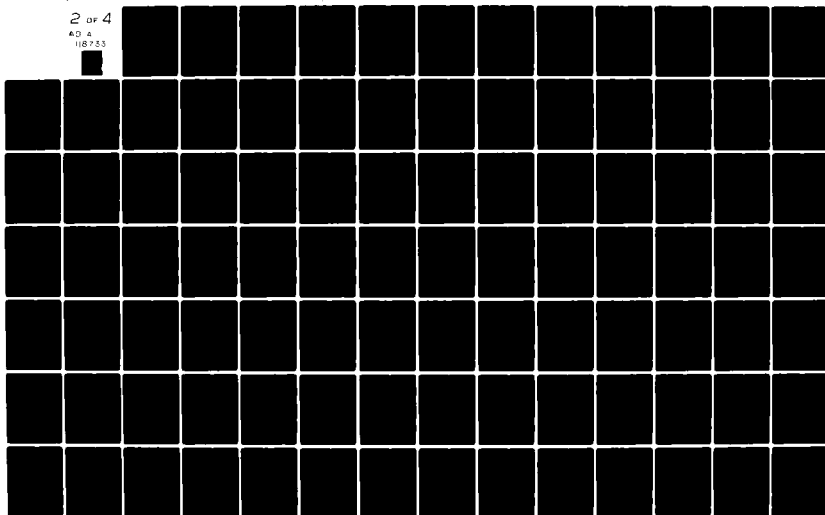


Figure 3-20 shows the resulting logic tree. A logic node has been created for each of the three regions on the scatter plot in Figure 3-19. The logic tree display shows which classes are present at each of the newly created nodes. Logic must be designed at each of these nodes because they all have more than one class present.

Classes "Whea" and "whea" are present at logic node 4. Since previous analysis shows that these classes do not overlap, a between-group logic design algorithm could be used to separate these two classes. Also, since there are only two classes present at logic node 4, the logic design process could be simplified by the use of a one-space algorithm.

Figure 3-21 shows a micro plot (top) of an eigenvector projection produced by the command L1EIGV. A user-drawn boundary partitions the display into two regions. The user has specified that the left region contains the class "Whea", and the right region contains the class "whea". The confusion matrix display at the bottom of Figure 3-21 shows the results of a partial evaluation at logic node 4. All vectors at the logic node were correctly classified. For each region in the micro plot, a logic node was created (See bottom of Figure 3-23B). Note that each of the newly created nodes has only one class present. Therefore, logic design along this path of the tree is complete.

OLPARS LOGIC TREE 'GRAINLOG'
CLASSES PRESENT: SscowhCkqr

DATE: 28-JAN-82 22:27:46

| | |
|-----|---------|
| 2 | INCMPLT |
| Ckq | |

| | |
|------------|--------|
| 1 | L2EIGU |
| SscowhCkqr | |

| | |
|-------|---------|
| 3 | INCMPLT |
| Sscor | |

| | |
|----|---------|
| 4 | INCMPLT |
| wh | |

DESIGN DATA SET: NGRAINS (****)
CLASS COUNT: 10 DIMENSIONALITY: 7

CURRENT
OPTION:
L2EIGU

CDISPLAY
CREATLOG
CSCALE
DBNDY
DRAWBNDY
PROJMN
PRTIDX
RDISPLAY
REDRAW
REPROJECT
SCALZM
SCALRET
SELECT

DATASET:
NGRAINS

Figure 3-20 Logic Tree After Between-Group Logic Generation

MICRO PLOT (MEAS. 1) DATE: 28-JAN-82 22:46:00
 * W * W GLOBAL SCALE
 VECTOR COUNTS

W

16

12

8

4

BINWIDTH
 CDISPLAY
 CREATLOG
 CSCALE
 DBNDY
 DRAWBNDY
 INTENSIFY
 PROJMM
 PRTIDX
 RDISPLAY
 REDRAW
 REPROJECT
 SCALZH
 SCALRET
 SELECT

(+)+1 (X)2

MIN = 4.0465E+02 BIN SIZE = 1.131E+01
 MAX = 4.3859E+02 BINS = 3

CURRENT
 OPTION:
 LIEIGU

DATASET:
 NGRAINS

CONFUSION MATRIX (BTWN. GROUP LOGIC) DATE: 28-JAN-82 22:47:03

REGION LOGIC DISPLAY SYMBOLS OF LOGIC NAME- GRAINLOG

LEFT 16 W
 RIGHT 15 W

| CLASS NAMES | LOGIC (PARENT) | NODES (CHILDREN) | | SUMS AND PERCENTAGES | | | |
|-------------|----------------|------------------|-------|----------------------|-------|---------|-------|
| | | 16 | 15 | (CORRECT) | | (ERROR) | |
| | | | | COUNT | PRCNT | COUNT | PRCNT |
| whea | 21 | 0 | 21 | 21 | 100.0 | 0 | 0.0 |
| Whea | 20 | 20 | 0 | 20 | 100.0 | 0 | 0.0 |
| TOTAL | 41 | 20 | 21 | 41 | 100.0 | 0 | 0.0 |
| CORRECT | | 20 | 21 | | | | |
| (PRCNT) | | 100.0 | 100.0 | | | | |
| ERRORS | | 0 | 0 | | | | |
| (PRCNT) | | 0.0 | 0.0 | | | | |

RESULTS OK (Y/N)? y

BINWIDTH
 CDISPLAY
 CREATLOG
 CSCALE
 DBNDY
 DRAWBNDY
 INTENSIFY
 PROJMM
 PRTIDX
 RDISPLAY
 REDRAW
 REPROJECT
 SCALZH
 SCALRET
 SELECT

CURRENT
 OPTION:
 LIEIGU

DATASET:
 NGRAINS

Figure 3-21 Creating Between-Group Logic on 1-Space Display

3.10.3 "Within-Group" Logic -

In order to complete the logic tree, logic must be designed at nodes 2 and 3. Since both of these nodes have overlapping classes, complete within-group algorithms should be chosen to design the logic.

Logic node 2 has three classes present: "alfa", "klov", and "Clvr". The confusion matrix display in Figure 3-19 shows (from these three classes), a total of 80 vectors correctly assigned to logic node 2. A class count of 3 and a vector count of 80 seems small in comparison with the total design data set of 10 classes and 293 vectors. Therefore, based on the guideline in Figure 3-17, it is reasonable to use the nearest mean vector algorithm to design logic at this node.

Logic node 3 has five classes present: "corn", "oats", "rye", "Soy", and "soys". The confusion matrix display in Figure 3-19 shows, from these five classes, a total of 170 vectors assigned to logic node 3. Even though the class and vector counts seem relatively large (half the classes and more than half the vectors from the total design data set), the Fisher pairwise logic design algorithm can be used at this node. (Note, class "rye" could be separated from the other classes by data partitioning).

The confusion matrix display at the top of Figure 3-22 shows the results of a partial evaluation at logic node 2 after Nearest Mean Vector logic was designed at the node. The logic was designed using the "weighted vector" distance option with no reject distances (see the NMV command description). The results of the evaluation show that 79 out of 80 vectors (98.75 percent) were correctly classified.

The confusion matrix display at the bottom of Figure 3-22 shows the results of a partial evaluation at logic node 3 after Fisher pairwise logic was designed at the node. The logic was designed using one threshold and a minimum vote count of one (see the FISHER command description). The results of the evaluation show that 169 out of 170 vectors (99.41 percent) were correctly classified.

Figures 3-23A and 3-23B show the completed logic tree GRAINLOG. Notice that all lowest nodes are either reject nodes or have only one class present.

3.10.4 Testing The Logic (Overall Logic Evaluation) -

In Section 3.6 the procedure for creating both a design data set and a test data set was described. The top of Figure 3-24 shows the test set GRAINS2 which was created before beginning structure analysis.

PARTIAL NEAREST MEAN VECTOR EVALUATION FOR LOGIC MODE 2
 LOGIC NAME: GRAINLOG DESIGN DATA SET NAME: NGRAINS (****)
 DIMENSIONALITY - 7

MMUMOD
 PRTCM
 RDISPLAY
 SUMMCM

ASSIGNED CLASSES

| | Clvr | klov | alfa |
|------|------|------|------|
| Clvr | 21 | 0 | 0 |
| klov | 0 | 22 | 0 |
| alfa | 0 | 1 | 36 |

TRUE CLASSES

| | Clvr | klov | alfa |
|------|-------|-------|------|
| TOTL | 21 | 22 | 37 |
| CORR | 21 | 22 | 36 |
| PRCT | 100.0 | 100.0 | 97.3 |
| EROR | 0 | 0 | 1 |
| PRCT | 0.0 | 0.0 | 2.7 |
| REJT | 0 | 0 | 0 |
| PRCT | 0.0 | 0.0 | 0.0 |

TOTAL NUMBER OF VECTORS = 80
 OVERALL CORRECT 79 FOR 98.75 PRCT
 OVERALL ERROR 1 FOR 1.25 PRCT
 OVERALL REJECT 0 FOR 0.00 PRCT

PRINTOUT (Y/N)? N
 ERROR LISTING (Y/N)? N

CURRENT
 OPTION:
 NMEVAL

DATASET:
 NGRAINS

PARTIAL PAIRWISE EVALUATION FOR LOGIC MODE 3
 LOGIC NAME: GRAINLOG DESIGN DATA SET NAME: NGRAINS (****)
 DIMENSIONALITY - 7

FISHMOD
 OPTIMLMOD
 PRTCM
 RDISPLAY
 SUMMCM
 THRESHMOD

ASSIGNED CLASSES

| | Soy | soys | corn | oats | rye |
|------|-----|------|------|------|-----|
| Soy | 20 | 0 | 0 | 0 | 0 |
| soys | 0 | 22 | 0 | 0 | 0 |
| corn | 0 | 0 | 41 | 0 | 0 |
| oats | 0 | 0 | 0 | 45 | 0 |
| rye | 0 | 0 | 0 | 1 | 41 |

TRUE CLASSES

| | Soy | soys | corn | oats | rye |
|------|-------|-------|-------|-------|------|
| TOTL | 20 | 22 | 41 | 45 | 42 |
| CORR | 20 | 22 | 41 | 45 | 41 |
| PRCT | 100.0 | 100.0 | 100.0 | 100.0 | 97.6 |
| EROR | 0 | 0 | 0 | 0 | 1 |
| PRCT | 0.0 | 0.0 | 0.0 | 0.0 | 2.4 |
| REJT | 0 | 0 | 0 | 0 | 0 |
| PRCT | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

TOTAL NUMBER OF VECTORS = 170
 OVERALL CORRECT 169 FOR 99.41 PRCT
 OVERALL ERROR 1 FOR 0.59 PRCT
 OVERALL REJECT 0 FOR 0.00 PRCT

PRINTOUT (Y/N)? Y
 ERROR LISTING (Y/N)? N

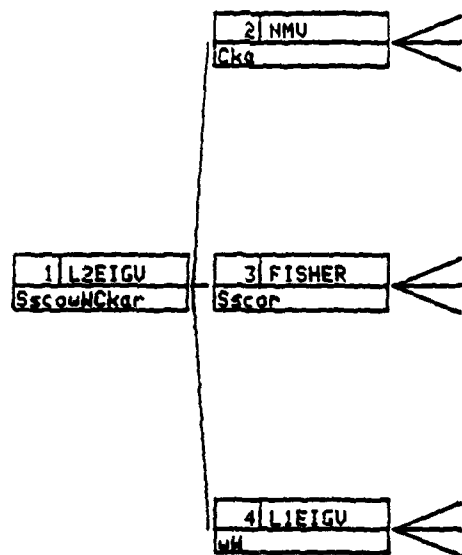
CURRENT
 OPTION:
 PNEVAL

DATASET:
 NGRAINS

Figure 3-22 Within-Group Confusion Matrices

OLPARS LOGIC TREE 'GRAINLOG'
CLASSES PRESENT: SscowNckar

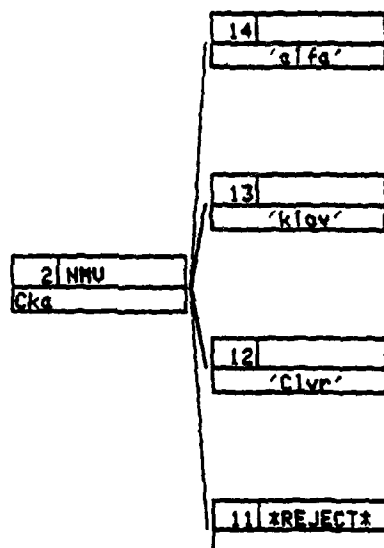
DATE: 01-FEB-02 11:19:42



DESIGN DATA SET: NGRAINS (****)
CLASS COUNT: 10 DIMENSIONALITY: 7

OLPARS LOGIC TREE 'GRAINLOG'
CLASSES PRESENT: SscowNckar

DATE: 01-FEB-02 11:20:05

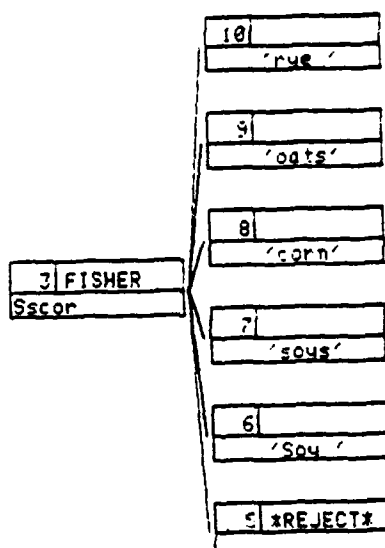


DESIGN DATA SET: NGRAINS (****)
CLASS COUNT: 10 DIMENSIONALITY: 7

Figure 3-23A 'Completed' Logic Tree

OLPARS LOGIC TREE 'GRAINLOG'
CLASSES PRESENT: SscowWckar

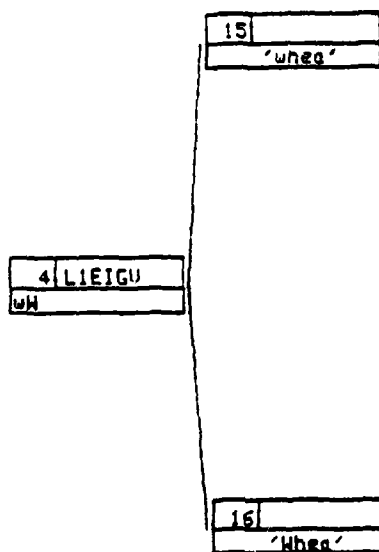
DATE: 01-FEB-82 11:20:26



DESIGN DATA SET: NGRAINS (****)
CLASS COUNT: 10 DIMENSIONALITY: 7

OLPARS LOGIC TREE 'GRAINLOG'
CLASSES PRESENT: SscowWckar

DATE: 01-FEB-82 11:20:46



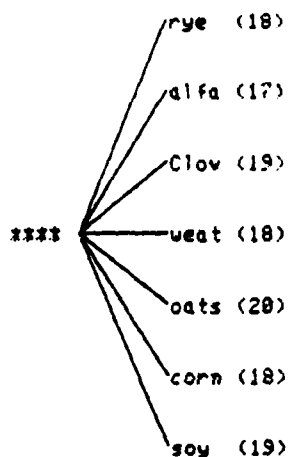
DESIGN DATA SET: NGRAINS (****)
CLASS COUNT: 10 DIMENSIONALITY: 7

Figure 3-23B 'Completed' Logic Tree

OLPARS DATA TREE 'GRAINS2'

DATE: 24-FEB-82 13:44:33

ANYTHING



NUMBER OF MEASUREMENTS PER VECTOR 12

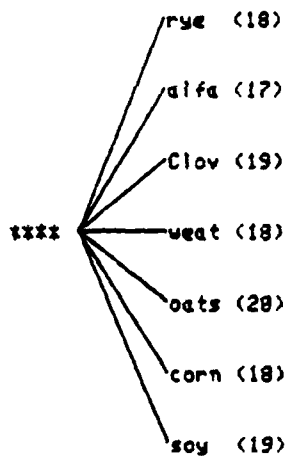
CURRENT
OPTION:
ANYTHING

DATASET:
GRAINS2

OLPARS DATA TREE 'graintst'

DATE: 28-JAN-82 23:10:45

RANK
SLCTMEAS
TRANSFRM
UNION



NUMBER OF MEASUREMENTS PER VECTOR 7

CURRENT
OPTION:
OSCRMEAS

DATASET:
graintst

Figure 3-24 Test Set Data Trees

You may recall that the design data set for the logic tree GRAINLOG is NGRAINS. NGRAINS was created by reducing the dimensionality of the original design data set GRAINS from 12 to 7. Since the design data set of GRAINLOG has a dimensionality of 7, any data set used to evaluate the logic must also have a dimensionality of 7. Thus, it will be necessary to reduce the dimensionality of GRAINS2 from 12 to 7.

The bottom of Figure 3-24 shows a newly created tree called GRAINTST, which will be used to evaluate the logic tree GRAINLOG. GRAINTST was created in the same manner as NGRAINS. The DSCRMEAS command was used to produce a rank order display. Measurements 1, 4, 8, 9, 10, 11, and 12 were selected, and a measurement reduction transformation was performed.

There is an alternate method for eliminating measurements from the logic design/evaluation process. Rather than reduce the design data set dimensionality, the user may, at each logic design stage, type in measurements to be eliminated (see Section 2.1.4). The measurements specified are then ignored during mathematical computations. The dimensionality of the design data set, however, remains the same. Logic evaluation can therefore proceed without having to reduce the test data set. This method of eliminating measurements may be useful when a complete within-group logic is the only logic being designed on a data set. In cases where one or more group logics are used in the design process, this method can

be cumbersome because the user must keep track of the measurements which are to be eliminated at each logic design stage.

3.10.5 Reassociated Names -

Utilizing the overall logic evaluation command, LOGEVAL, any data set may be tested against logic designed on any other data set of equal dimensionality. During the evaluation, vectors from the test data set are assigned to a lowest node of the logic tree. For completed logic trees, each lowest node (excluding reject nodes) has associated with it a design data set class name. If the data class name for a given vector matches the name associated with the logic node to which the vector was assigned, the vector is considered to be correctly classified. The totals and percentages of correctly classified vectors listed in the confusion matrix display will only be useful if the names of the data classes on which logic was designed are the same as the names of the data classes being evaluated. In cases where the logic names are not the same as the test data set names, the command REASNAME may be used to associate new names with the logic nodes, these new names are called reassociated names.

If reassociated names have been added to the logic tree, LOGEVAL asks the user whether or not the reassociated names are to be used during the evaluation. If the response is yes, the reassociated names will be used in place of the original design names to determine if the vectors in the data set being evaluated have been assigned correctly.

USING OLPARS -- 3
LOGIC DESIGN AND EVALUATION

As mentioned previously, the GRAINTST data set will be used to evaluate the logic tree GRAINLOG. The class names in GRAINTST are identical to the class names in GRAINLOG's design data set before it was restructured into subclasses. Because the original data set was restructured, before logic was designed, the classes "Clov", "weat", and "soy" which exist in GRAINTST are not represented in the logic tree GRAINLOG. Therefore, logic nodes in GRAINLOG which are currently associated with subclass names (from the restructuring process) should be reassociated with the original class names.

Figure 3-25 is a table of logic node numbers, original design data set class names, and reassociated class names for the logic tree GRAINLOG. The table was produced by the REASNAME command. Notice that logic nodes 16 and 15, associated with classes "whea" and "whea" respectively, have been reassociated with the class name "weat"; logic nodes 6 and 7, associated with "Soy" and "soys", respectively, have been reassociated with the class name "soy", and logic nodes 12 and 13 associated with classes "clvr" and "klov" respectively, have been reassociated with the class name "Clov". During an overall logic evaluation, these reassociated class names will be used to determine if the vectors in GRAINTST have been correctly classified.

| LOGIC NODE | DESIGN DATA SET CLASS NAME | REASSOCIATED CLASS NAME | ANYTHING |
|------------|-------------------------------|----------------------------|----------|
| 16 | whea | weat | |
| 15 | whea | weat | |
| 6 | soy | soy | |
| 7 | soys | soy | |
| 8 | corn | corn | |
| 9 | oats | oats | |
| 10 | rye | rye | |
| 12 | Clvr | Clcv | |
| 13 | Klov | Clcv | |
| 14 | alfa | alfa | |

MORE CHANGES (Y/N)? N

CURRENT
OPTION:
ANYTHING

DATASET:
GRAINTST

Figure 3-25 Viewing Reassociated Class Names

3.10.6 Accepting Or Rejecting A Logic Design -

Figure 3-26 is a confusion matrix display which summarizes the results of an overall logic evaluation in which the GRAINTST data set was tested against the GRAINLOG logic. The results show that 125 out of 129 vectors (96.90 percent) were correctly classified, 4 vectors (3.10 percent) were incorrectly classified, and no vectors were rejected.

At this point, based on the confusion matrix statistics, the analyst decides whether or not the logic is good enough to use in the design of a classification system for the particular type of data under consideration. As an analyst, you may want to design several different logics and compare the evaluation results in order to decide which logic is most suitable. There will probably be certain criteria that the logic must meet in order for it to be considered a viable solution to a particular problem. Some factors to consider in deciding whether to accept or reject a particular logic is discussed in the following text.

(1) Speed versus number of errors -

For any logic, the amount of time it takes to classify a vector is related to the dimensionality for the vector and the number of data comparisons that must be made (i.e., the complexity of the logic). In general, the more complex a logic is, and the greater the dimensionality of the vectors being classified, the longer the classification will take. On the other hand, less complex logics and a smaller vector dimensionality tend to produce more errors. Thus, there is a tradeoff in terms of how fast the answer is produced, and the probability that the answer is correct, therefore, you must determine which factor, speed or number of errors is more relevant to your particular problem.

OVERALL EVALUATION

LOGIC NAME - GRAINLOG
 DESIGN DATA SET NAME - NGRAINS (****)
 CURRENT DATA SET NAME - grainstst (****)
 DIMENSIONALITY - 7

ASSIGNED CLASSES

| | soy | soy | corn | oats | weat | weat | Clov | Clov | alfa | rye | RJCT |
|------|-----|-----|------|------|------|------|------|------|------|-----|------|
| soy | 9 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| corn | 0 | 0 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| oats | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| weat | 0 | 0 | 0 | 0 | 10 | 8 | 0 | 0 | 0 | 0 | 0 |
| Clov | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 10 | 2 | 0 | 0 |
| alfa | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 0 | 0 |
| rye | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 17 | 0 |

OVERALL EVALUATION

LOGIC NAME - GRAINLOG
 DESIGN DATA SET NAME - NGRAINS (****)
 CURRENT DATA SET NAME - grainstst (****)
 DIMENSIONALITY - 7

| TRUE CLASS | soy | corn | oats | weat | Clov | alfa | rye |
|---------------|-------|-------|-------|-------|------|-------|------|
| TOTL | 19 | 18 | 20 | 18 | 19 | 17 | 18 |
| CORR | 19 | 18 | 20 | 18 | 17 | 17 | 17 |
| PRCT | 100.0 | 100.0 | 100.0 | 100.0 | 89.5 | 100.0 | 94.4 |
| EROR | 0 | 0 | 0 | 0 | 2 | 0 | 1 |
| PRCT | 0.0 | 0.0 | 0.0 | 0.0 | 10.5 | 0.0 | 5.6 |
| REJT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PRCT | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

TOTAL NUMBER OF VECTORS = 129
 OVERALL CORRECT 126 FOR 97.67 PRCT
 OVERALL ERROR 3 FOR 2.33 PRCT
 OVERALL REJECT 0 FOR 0.00 PRCT

Figure 3-26 An Overall Logic Evaluation

(2) What is the effect (cost) of an error? -

In the logic evaluation example previously described, one vector from class "corn" was classified as "Clov" (See Figure 3-19). The analyst needs to determine the importance of such an error if it occurs in a real world situation. For certain data problems, an error may not be critical. If, for instance, "corn" and "Clov" were both going to receive the same fertilization treatment after classification, the fact that a group of "corn" vectors are classified as "Clov" is insignificant. If, however, the two classes were to receive different fertilization treatments, and the treatment applied to "Clov" crops would damage "corn" crops, then incorrectly classifying a group of "corn" vectors would have more serious consequences.

As another example, consider the problem of separating "sick" people from "healthy" people. To classify a healthy person as sick may not be a serious mistake, but to classify a sick person as healthy may prevent that person from seeking medical attention. Clearly, the consequences of such an error could be very grave.

(3) Rejecting vectors versus misclassifying them -

In order to minimize the probability of occurrence of certain types of errors in the real world, the analyst may decide that these errors should not be allowed to occur in the logic design/evaluation process. Certain errors may sometimes be eliminated by specifying (or increasing the size of) reject regions. In effect, a reject region mathematically defines an area surrounding a class. In order for vectors to be assigned to a class, they must fall within the reject region. Vectors which do not fall within the reject region of a class (to which they would otherwise be assigned) are classified as "unknown" objects.

The advantage of specifying reject regions is that it reduces the number of errors in the logic. The disadvantage, however, is that some vectors, which would have been correctly classified without specifying reject regions, may be rejected. Consequently, you must decide which set of circumstances is best for a particular data problem; rejecting "known" objects, or misclassifying "unknown" vectors.

3.10.7 Viewing Your Logic -

As an OLPARS user, you can obtain a printed listing of the decision logic via the command PRTLOG. The logic can be used to program the corresponding decision algorithm to be used for pattern classification. Remember, OLPARS is not a pattern classification system; rather it is a research tool which is used to design pattern classification systems.

3.11 SUMMARY

Section 3 of the OLPARS user manual has been designed to introduce a new user to the capabilities available in OLPARS. Following is a brief summary of the procedures described; analysis of data, evaluation of measurements, data transformations, and logic design and testing.

The user collects data which represents a "real world" environment. The data is translated into a vector format which can be interpreted by OLPARS. Each vector component is a feature in the environment. Each vector represents an object in the environment and belongs to a data class, which is considered an environmental state. The user attempts to define features which will yield information to aid in discriminating between the various environmental states (feature extraction).

USING OLPARS -- 3

SUMMARY

Once the data is in the "proper" format, the user may "log in" to OLPARS and create a data tree. Remember, some time before designing logic, this original data set must be divided into two new data sets, one for designing logic, and one for evaluating (testing) the logic (Note, this step is not necessary if a test set has been collected separately from the design set).

The next step, after creating a data tree, is to begin analysis of the data. Initially, if the data vectors are not comprised of true features, but "raw" measurements, it is desirable to examine the statistical properties of the data to help derive useful feature vectors. The statistical information that can be viewed consists of data ranges and measurement overlap, data means, standard deviations, covariance matrices, and correlation matrices. This type of analysis could be useful for identifying data collection or translation errors. Also, the ranges and overlap graphs are useful in identifying features which distinguish classes from one another.

Following a visual examination of the raw measurements and statistics, the structure of the data can be studied using the OLPARS structure analysis commands. The purpose of structure analysis is to search for clusters in the data and to identify multimodal classes. If multimodal classes are found to exist, they are divided (restructured) into subclasses.

If the data set is in excess measurement mode (see Section 2.1.3), or if the user wishes to reduce the vector dimensionality of the data set in order to speed up OLPARS calculations, the measurement evaluation commands can be used to determine and select the "best" class discriminating measurements, and to transform the original data set into a new one containing only the "selected" measurements. If the data set is in excess measurement mode, measurement evaluation will most likely precede structure analysis, because only the coordinate projection commands operate in excess measurement mode.

Other data transformations available to an OLPARS user include normalization, eigenvector, and measurement transformations. The normalization transformation is used essentially to rescale the data; the eigenvector transformation is used to transform the original vectors into a user-defined eigenvector subspace; and the measurement transformation is used to define new features by combining or altering the original features. Both the eigenvector and the measurement transformations can be used to reduce the dimensionality of the data set.

After performing structure analysis and/or measurement evaluation/transformation on the data, the user is ready to design the recognition logic which will classify the various states of the environment, using the selected features of the design data set. Basically, the between-group logic commands should be used to separate non-overlapping classes, and the complete within-group logic commands should be used to separate statistically overlapping

USING OLPARS -- 3
SUMMARY

classes.

When the logic design process is complete, the logic can be evaluated using the test data set. Remember that any measurement transformation/reduction that was performed on the design data set must also be applied to the test data set. Also, before an overall logic evaluation is performed, each subclass should be re-identified with the original multimodal classes by using the "reassociated names" (REASNAME) command.

Lastly, based on the evaluation results, the user must decide whether the logic should be accepted or rejected for use in the design of a classification system. If the user accepts the logic, it can be printed at the lineprinter. If the user rejects the logic, it may be necessary to reiterate the entire procedure, using different features and/or different OLPARS algorithms, in an attempt to design "better" logic.

It is important to realize that the analysis/logic design procedure for the data set GRAINS, described previously, is intended to be an example (guideline) for a novice OLPARS user. It is not the only way to design pattern classification logic. Once you become familiar with your data, and the OLPARS algorithms, you can develop your own methods for creating logic.

SECTION 4

OLPARS COMMANDS

4.0 GENERAL STRUCTURE

The command structure represents the primary interface between OLPARS and the user. OLPARS consists basically of a filing system and a collection of individual programs that operate on data in the files to perform pattern recognition functions. Often there are several commands (or algorithms) for performing a particular function; for a particular set of data, one program may prove more satisfactory than another. Also, certain functions may be accomplished by executing a sequence of commands in which, after the first command is completed, the user must select the most appropriate "next-command" (from a list of meaningful options) based on his/her data and problem being investigated.

Thus it should be clear that OLPARS requires a user who understands the principles of pattern recognition, the mathematical significance of the various programs to be chosen, the data being evaluated, and the problem. What this user requires of OLPARS is a system with a command structure which provides simplicity of operation, responsiveness, and flexibility.

OLPARS COMMANDS -- 4 GENERAL STRUCTURE

The command structure of "portable" OLPARS is completely flexible and structurally free; that is, when the system is expecting an OLPARS command, any command may be entered -- there is no forced hierarchical structure imposed. This freedom, however, is often more apparent than real because it is only meaningful to select certain commands at particular points in the pattern recognition process. (The system design includes a feature of presenting the user with a list or menu of commands that are the most reasonable to use after the completion of any individual command).

Since OLPARS consists of a large number of commands, it is quite natural in a discussion of the commands and command structure to categorize the commands into groups. The categorization is somewhat arbitrary because of their variety, the ways in which they may be combined in sequences, and the cross usage of some commands in different pattern recognition functions.

In the categorization scheme used here, there are two basic divisions of commands: Utility Commands and Analytic commands. The Utility category has three subcategories: Data Manipulation, Display Manipulation, and Information. These commands are self-contained entities, each independent, and are used in any order which the user determines to be appropriate.

The Analytic commands consist of programs which implement the principal pattern recognition algorithms of OLPARS. This category has four subcategories: Measurement Evaluation, Transformations,

Structure Analysis, and Logic Design. The Logic Design category is further subdivided into two groups, between-group logic commands and within-group logic commands.

An important distinction between the commands in the Utility category and the Analytic category is that most Analytic commands have an appropriate set of subsidiary commands associated with them. These subsidiary commands implement various options related to a particular Analytic command, and as a result, there is the implied hierarchy of commands; that is, it is only meaningful to call a subsidiary command after its related Analytic command has been executed.

Figure 4-1 is a schematic illustration of the command categorization as previously mentioned. Figure 4-2 is a list of all currently implemented OLPARS commands in their respective categories. The following section gives an alphabetically ordered list along with a brief description of each of the commands.

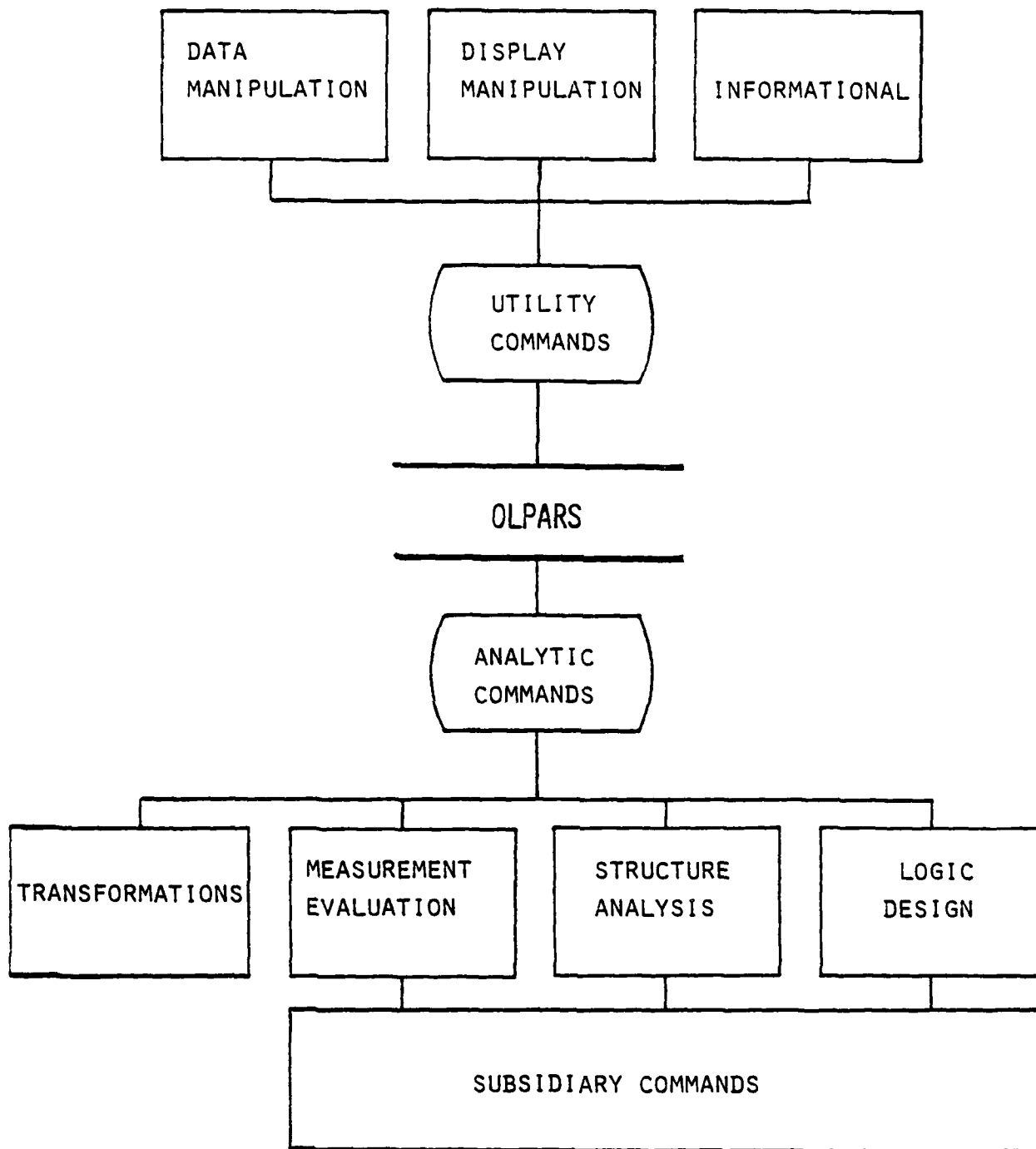


FIGURE 4-1 COMMAND CATEGORIES IN OLPARS

Currently implemented OLPARS functions

| Logic Design | Structure Analysis | (data) | -Utilities- (display) | (info.) |
|--------------|--------------------|--------------|--------------------------|-----------|
| ===== | ===== | ===== | ===== | ===== |
| (b-group) | | (data tree) | (1-space) | ANYTHING |
| L1ASDG | | APPEND | BINWIDTH | BYEOLP |
| L2ASDG | | COMNOD | INTENSIFY | CDEFAULT |
| L1CRDV | *S1CRDV | CRANDTS | | DTRENAME |
| L2CRDV | *S2CRDV | DDATANOD | (1,2-pace) | HELP |
| L1EIGV | S1EIGV | DDATATREE | CDISPLAY | LISTLOGS |
| L2EIGV | S2EIGV | DDSUBSTR | CSCALE | LISTTREES |
| L2FSHP | S2FSHP | DRAWTREE | DBNDY | LTRENAME |
| CREATLOG | RESTRUCT | DVEC | DRAWBNDY | MATRIX |
| | | *FILEIN | PROJMN | PRTCM |
| (w-group) | Meas. Eval. | *FILEOUT | PRTIDX | *PRTDS |
| FISHER | & Transform | MAKETREE | REDRAW | PRTLOG |
| FISHMOD | ===== | MOVEC | REPROJECT | RDISPLAY |
| OPTIMLMOD | *DSCRMEAS | | SCALRET | SETDS |
| THRESHMOD | *RANK | (logic tree) | SCALZM | SETLOG |
| PWEVAL | *SLCTMEAS | DLOGTREE | SELECT | SUMMCM |
| | *TRANSFORM | DLSUBSTR | | |
| NMV | *UNION | DRAWLOG | | |
| NMVMOD | | NAMELOG | | |
| NMEVAL | EIGNXFRM | REASNAME | | |
| | MATXFRM | | | |
| NRSTNBR | *MEASXFRM | | * can be used in | |
| NNMOD | NORMXFRM | | Excess Measurement Mode | |
| LOGEVAL | | | | |

Figure 4-2 OLPARS Command Categories

OLPARS CCMANDS -- 4
CCMMAND SUMMARY

4.1 COMMAND SUMMARY

- ANYTHING - Display available OLPARS commands.
- APPEND - Append a data tree node from one tree to another tree.
- BINWIDTH - Alter the bin size of a one-space display.
- CDEFAULT - Change default setting of CLPAR system entities.
- CDISPLAY - Change the current one-space macro display to a one-space micro display, or vice versa

-OR-

Change the current two-space cluster plot to a two-space scatter plot, or vice versa.

- CCMNOD - Combine two or more lowest nodes in a data tree.
- CRANDTS - Create a random data test set.
- CREATLOG - Create "between-group" logic for one-space or two-space projections.
- CSCALE - Change the scaling of a one-space/two-space display from "square" to "rectangular" and vice-versa.
- DBNDY - Delete existing boundaries drawn on a one-space/two-space display.
- DDATANOD - Delete a lowest node from a data tree.
- DDATATREE - Delete a data tree from user directory.
- DDSUBSTR - Delete data tree substructure.
- DLOGTREE - Delete a logic tree from user directory.
- DLSUBSTR - Delete logic tree substructure.
- DRAWBNDY - Partition a one-space/two-space projection for data restructuring or "between-group" logic generation.
- DRAWLOG - Display the structure of a logic tree.
- DRAWTREE - Display the structure of a data tree.

OLPARS COMMANDS -- 4
COMMAND SUMMARY

| | |
|-----------|---|
| DSCRMEAS | - Compute measurement evaluation statistics and present an "overall" ranking of measurement discriminating power. |
| DTRENAME | - Rename an OLPARS data tree. |
| DVEC | - Delete vectors from a data class. |
| EIGNXFM | - Transform a data set (creating a new data set) by projecting it into a subspace determined by a subset of its eigenvectors. |
| FILEIN | - Create an OLPARS data tree from a text file data vectors. |
| FILEOUT | - Place data tree vectors into a text file. |
| FISHER | - Create Fisher pairwise discriminant logic a node of a logic tree. |
| FISHMCD | - Modify the vote or threshold count of a logic node. |
| HELP | - Provide help on an OLPARS command or subject. |
| INTENSIFY | - Intensify or highlight user specified classes a one-space micro display. |
| L1ASDG | - One-space assigned discriminant group (on group defined Fisher discriminant) for logic design. |
| L1CRDV | - One-space coordinate projection for logic design. |
| L1EIGV | - One-space eigenvector projection for logic design. |
| L2ASDG | - Two-space assigned discriminant group projection (on plane defined by group determined Fisher discriminant and Fisher orthogonal) for logic design. |
| L2CRDV | - Two-space coordinate projection for logic design. |
| L2EIGV | - Two-space eigenvector projection for logic design. |
| L2FSHP | - Two-space Fisher discriminant projection (on plane defined by two Fisher discriminants; the Fisher discriminants are based on user specified class pairs) for logic design. |
| LISTLOGS | - Display user logic tree names. |
| LISTREES | - Display user data tree names. |
| LOGEVAL | - Evaluate a data set against a logic tree. |

CLPARS CCMANDS -- 4
CCMAND SUMMARY

| | | |
|-----------|---|---|
| LTRENAME | - | Rename an CLPARS logic tree. |
| MAKETREE | - | Create a data tree using nodes from existing data trees. |
| MATRIX | - | Utility for maintaining the saved transformation matrix file. |
| MEASXFRM | - | User specified data measurement transformation. |
| MOVEC | - | Move vectors in a two-space coordinate projection (meant for generating reference patterns for Nearest Neighbor logic). |
| NAMELOG | - | Create a new logic tree. |
| NMEVAL | - | Evaluate a Nearest Mean Vector logic node. |
| NMV | - | Create a Nearest Mean Vector logic node. |
| NMVMCD | - | Modify logic at a Nearest Mean Vector logic node. |
| NNMOD | - | Modify logic at a Nearest Neighbor logic node. |
| NORMXFRM | - | Transform a data set (creating a new data set) to "normalized" measurements by dividing each measurement by the overall standard deviation of that measurement. |
| NRSTNBR | - | Create a Nearest Neighbor logic node (and a Nearest Neighbor reference pattern tree). |
| OPTIMLMOD | - | Create/modify optimal discriminant logic at a Fisher logic node. |
| PROJMN | - | Display mean vector of projected data class on the one-space/two-space display. |
| PRTCM | - | Print confusion matrix. |
| PRTDS | - | Print data set vectors and statistics. |
| PRTIDX | - | Print vector identifiers and vector quantities. |
| PRTLOG | - | Print logic tree information. |
| PWEVAL | - | Evaluate a Fisher pairwise logic node. |
| RANK | - | Rank data measurements according to user selected ranking method. |
| RDISPLAY | - | Redisplay a one-space, two-space or confusion matrix display. |

| | |
|-----------|---|
| REASNAME | - Modify the reassociated class names in a logic tree. |
| REDRAW | - Display an existing one-space threshold or two-space boundary. |
| REPROJECT | - Choose different projection vectors for eigenvector projection displays already generated. |
| RESTRUCT | - Restructure (subdivide) one class in a data tree. |
| S1CRDV | - One-space coordinate projection for structure analysis. |
| S1EIGV | - One-space eigenvector projection for structure analysis. |
| S2CRDV | - Two-space coordinate projection for structure analysis. |
| S2EIGV | - Two-space eigenvector projection for structure analysis. |
| S2FSHP | - Two-space Fisher discriminant projection (on plane defined by two Fisher discriminants; the Fisher discriminants are based on user specified class pairs) for structure analysis. |
| SCALRET | - Return a one-space/two-space display to its original display scale. |
| SCALZM | - Change display scale of a one-space/two-space display by "zooming in" on a subset of the current display (a view magnification). |
| SELECT | - Select class symbols to be displayed on a one-space/two-space display. |
| SETDS | - Select the "current" data set. |
| SETLOG | - Select the "current" logic. |
| SLCTMEAS | - Manual measurement selection performed by user on a rank order display; used in subsequent data selection/tree transformation. |
| SUMMCM | - Display confusion matrix summary. |
| THRESHMOD | - Modify thresholds in Fisher logic node. |
| TRANSFRM | - Transform existing tree (creating a new tree) by using algorithmic or user specified "selected" measurements of a rank order display. |

OLPARS COMMANDS -- 4
COMMAND SUMMARY

UNION - Algorithmic selection of measurements in a rank order display for data set transformation; selects measurement(s) which are the "best" discriminators for any class or class pair.

4.2 COMMAND DESCRIPTIONS

This section contains descriptions of commands that can be executed in OLPARS. The general format of the command description is given in Figure 4-3. If a particular item of the format does not pertain to a command (e.g., an example is not given because there is no user interaction), the item is omitted from the description or is followed by the word "NONE".

COMMAND NAME:

The name used to activate the command

CATEGORY:

The command categorization types:

- Utility command
- Measurement Evaluation command (subsidiary)
- Structure Analysis command (subsidiary)
- Logic Design command (subsidiary)
- Transformation command

FUNCTIONAL DESCRIPTION:

A functional description of the command

USER INTERACTION:

Program requests, responded to by the user

EXAMPLE(S):

One or more examples of running the command at a terminal are given here. Note: text between slashes denotes user responses. The OLPARS standard exit is represented by /<CR>/ immediately following a user prompt, unless otherwise noted.

Session with SHORT prompts

User prompts that occur when the prompt flag in the CM file indicates "short prompts" are shown here.

(PROMPT NOTES: *****
A further explanation of user interaction is given here. The reader is informed of any errors that may occur as a result of an incorrect response to a prompt or an OLPARS filing system error. Anything extra that the user might need to know is explained here.

Session with LONG prompts

User prompts that occur when the prompt flag in the CM file indicates "long prompts" are shown here.

Figure 4-3 OLPARS Command Description Format

OLPARS COMMANDS -- 4
CCMMAND DESCRIPTIONS

OLPARS COMMAND DESCRIPTIONS

ANYTHING

COMMAND NAME: ANYTHING

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

ANYTHING displays the names of all the currently
available CLPARS commands.

USER INTERACTION: NONE

APPEND

COMMAND NAME: APPEND

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

APPEND adds one node from a source tree to another tree (or the same tree).

USER INTERACTION:

The user is asked for the node names of two trees(the names may be the same). The first name the user gives to the program specifies the source tree (the tree from which the new node is to be obtained). The second name specifies the receiving tree (the tree to which the node is to be appended).

The user is then requested for the names of two nodes. The first node will be the source node, the second will be the receiving node.

Next the user is asked for a new name for the node that is being appended. Lastly , the user is asked if the vector identifiers of the appended node are to be resequenced.

EXAMPLE(S):

In the following examples GRAIN1 and GRAIN2 are data trees. GRAIN1 has nodes RYE, CORN, CLOV, and WHEA all under the senior node. GRAIN2 has nodes ALFA, and CORN. The node RYE will be appended to GRAIN2 under the senior node.

Session with SHORT prompts

SOURCE TREE? /GRAIN1/

RECEIVING TREE? /GRAIN2/

SOURCE NODE? /RYE /

RECEIVING NODE? /****/

NEW NAME? /CORN/

NOT A UNIQUE NAME
NEW NAME? /RYE/

APPEND (continued)

RESEQUENCE VECTOR IDENTIFIERS(Y,N)? /N/

(PROMPT NOTES: *****)

The source node that is entered by the user must be a lowest node. The receiving node entered by the user must be an intermediate node. If not, proper error messages will be displayed. As shown above, the new name that is entered must be unique. Also, the display symbol must be unique.

*****)

Session with LONG prompts

ENTER THE NAME OF THE SOURCE TREE
(MAXIMUM 8 CHARACTERS)? /GRAIN1/

ENTER THE NAME OF THE RECEIVING TREE
(MAXIMUM 8 CHARACTERS)? /GRAIN2/

ENTER THE NAME OF SOURCE NODE
(MUST BE A LOWEST NODE, MAX 4 CHARACTERS)? /RYE/

ENTER THE NAME OF THE RECEIVING NODE
(MUST BE INTERMEDIATE NODE, MAX 4 CHAR)? /****/

ENTER A NEW NODE NAME OF THE NODE BEING APPENDED
(MAXIMUM 4 CHARACTERS)? /RYE/

DO YOU WANT TO RESEQUENCE THE VECTOR IDENTIFIERS
IN THE NODE YOU ARE APPENDING(Y,N)? /N/

BINWIDTH

COMMAND NAME: BINWIDTH

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

BINWIDTH will first determine whether the current display is a valid one-space display. If it is a one-space display, the scale is modified by either changing the starting point, the number of bins, and/or the interval size.

USER INTERACTION:

You are first asked if the minimum scale value is to be changed (the starting 'x' point of the display). Then you are asked if the number of bins is to be changed. Lastly you are asked if the interval (bin size) is to be changed. These can be answered in any way, all yes, all no, or any combination of the three. If you do not alter a scale value, its value will remain the same.

EXAMPLE(S):

In the following one-space prompt you will see the fullest amount of prompts possible. This is because yes ('Y') will be answered to all the change prompts. The number ten '10' will be entered as the new minimum scale value, '20' as the new number of bins, and '25' as the new binsize. The current minimum scale value will be 1, current number of bins will be 14, and the current binsize will be 58.34.

Session with SHORT prompts

CURRENT MINIMUM VALUE IS ** 1.000 **
CHANGE MINIMUM SCALE VALUE (Y/N)? /Y/
ENTER NEW POINT - /10/

CURRENT NUM. OF BINS IS ** 14 **
CHANGE NUMBER OF BINS (Y/N)? /Y/
BINS - /20/

CURRENT BINSIZE IS ** 58.34 **
CHANGE BINSIZE (Y/N)? /Y/
BINSIZE - /25/

(PROMPT NOTES: *****)

If you wish to change the number of bins, the number entered must be greater than zero. If a number less than zero is entered, a message is printed at the terminal, and you are prompted for another number.

If the minimum scale value is changed, the number of bins is not, and you wish to change the binsize, a message will be printed at the terminal with the smallest binsize that can be entered. If a number less than the given number is entered, the number of bins is set to the maximum number of bins (50), and the binsize is set to the number displayed at your terminal as the smallest allowable binsize.

If the classes are all grouped together near the minimum value, and you would like to see them spread out, you should:

- 1) allow the minimum scale value to remain the same
- 2) decrease the number of bins
- 3) decrease the binsize

BINWIDTH (continued)

The following is a table showing what happens to each value when you enter either 'yes' or 'no' to the prompt.

xmin - the minimum scale value
 xmax - the maximum scale value
 # bins - the total number of bins
 bin - the binsize (interval size)
 mod - this variable was recalculated by the computer
 ent - this variable has the value you entered
 same - variable value remains the same

| xmin | #bins | binsize | xmin | #bins | binsize | xmax |
|------|-------|---------|------|-------|---------|-------|
| ---- | ----- | ----- | ---- | ----- | ----- | ----- |
| Y | Y | Y | ent | ent | ent | mod |
| Y | N | N | ent | same | same | same |
| Y | Y | N | ent | ent | same | mod |
| Y | N | Y | ent | same | ent | mod |
| N | N | N | same | same | same | same |
| N | Y | Y | same | ent | ent | mod |
| N | Y | N | same | ent | same | mod |
| N | N | Y | same | same | ent | mod |

Formulas for finding the modified variables

mod xmax -----> (# bins * binsize) + xmin

*****)

BINWIDTH (continued)

Session with LONG prompts

CURRENT MINIMUM VALUE IS ** 1.000 **
DO YOU WANT TO CHANGE THE STARTING POINT OF THE
DISPLAY (MINIMUM SCALE VALUE) Y-N ? /Y/
ENTER THE NEW MINIMUM POINT
FOR THE DISPLAY - /10/

CURRENT NUM. OF BINS IS ** 14 **
DO YOU WANT TO CHANGE THE NUMBER OF BINS
IN DISPLAY (Y/N)? /Y/
ENTER THE NUMBER OF BINS FOR THE DISPLAY - /20/

CURRENT BINSIZE IS ** 58.34 **
DO YOU WANT TO CHANGE THE INTERVAL(BIN) SIZE(Y/N) /Y/
ENTER A NUMBER FOR THE NEW BINSIZE - /25/

BYEOLP

COMMAND NAME: BYEOLP

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

BYEOLP's sole purpose is to "logout" or exit OLPARS.
What other functions this command may perform is
left to the local implementation.

USER INTERACTION:

Left to local implementation.

COMMAND NAME: CDEFAULT

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

CDEFAULT first reads the current values from the Communications (CM) File and displays them for the user (cluster/scatter cut-off value, one-space bin factor, prompt flag, instrumentation flag and instrumentation threshold value). CDEFAULT then puts up a list of options and asks the user to select one. By selecting the appropriate option, the user may (1) change the cluster/scatter cut-off value, (2) change the one-space bin factor, (3) change from long prompts to short prompts or vice versa, (4) change the instrumentation flag from 'ON' to 'OFF' or vice versa, (5) change the instrumentation threshold value, or (6) set default values for the CM file parameters mentioned in 1,2,3,4 and 5 above. The effects or changes in these options are as follows:

- Option 1: If the number of vectors in a data set to be projected on a two-space plot is greater than the cut-off value, a cluster plot is displayed, otherwise a scatter plot is shown. If option 1 is selected, the program requests that a new value be input. Example: if the cut-off value were set to zero, all two-space plots would initially be displayed in the cluster mode.
- Option 2: The initial number of bins for a histogram plot is determined by dividing the total number of vectors by the product of the number of classes and the one-space bin factor. If option 2 is selected, the program requests that a new bin factor value be entered. Example: if the number of classes is 1, the bin factor is set to 2, and the total number of vectors is 100, these vectors will be placed in $100/(1*2) = 50$ bins.
- Option 3: The default value of the prompt flag is 0 - use short prompts. By changing to long prompts (flag = 1), the user receives longer messages requesting information.

CDEFAULT (continued)

- Option 4: The default value of the instrumentation flag is C - disable (turn off) instrumentation. By changing the instrumentation flag to 1, the instrumentation is enabled (turned on). (Note, with instrumentation turned on, program execution time will increase dramatically.)
- Option 5: The default value of the instrumentation threshold is 5. If instrumentation is 'ON', a program must have completed successfully at least 5 times in order to turn off the printing of debug information. (To force out all instrumentation, set the threshold to zero.)
- Option 6: If option 6 is selected, the cluster/scatter cut-off value is set to 500, the one-space bin factor is set to 5, the prompt flag is set to 0, the instrumentation flag is set to 0, and the instrumentation threshold is set to 5.

CDEFAULT continues to prompt the user for an option number until a carriage return (<CR>) is typed.

USER INTERACTION:

The user is asked to select an option number from the list of options. If option 1 is selected, the user is asked for the new cluster/scatter cut-off value. If option 2 is selected, the user is asked for the new one-space bin factor. If option 3 is selected, the user is asked for the new instrumentation threshold value.

EXAMPLE(S):

Session with SHORT prompts

CURRENT STATUS OF CM FILE PARAMETERS:
CLUSTER/SCATTER CUTOFF VALUE = 100
ONE-SPACE BIN FACTOR = 2
PROMPT FLAG = SHORT
INSTRUMENTATION FLAG = ON
INSTRUMENTATION THRESHOLD = 3

PROGRAM OPTIONS:

CHANGE:

1. CLUSTER/SCATTER CUTOFF VALUE
2. ONE-SPACE BIN FACTOR
3. PROMPT FLAG
4. INSTRUMENTATION FLAG
5. INSTRUMENTATION THRESHOLD

OR:

6. SET DEFAULT VALUES

OPTION NUMBER =/6/

CURRENT STATUS OF CM FILE PARAMETERS:

CLUSTER/SCATTER CUTOFF VALUE = 500

ONE-SPACE BIN FACTOR = 5

PROMPT FLAG = SHORT

INSTRUMENTATION FLAG = OFF

INSTRUMENTATION THRESHOLD = 5

PROGRAM OPTIONS:

CHANGE:

1. CLUSTER/SCATTER CUTOFF VALUE
2. ONE-SPACE BIN FACTOR
3. PROMPT FLAG
4. INSTRUMENTATION FLAG
5. INSTRUMENTATION THRESHOLD

OR:

6. SET DEFAULT VALUES

OPTION NUMBER =/<CR>/

(PROMPT NOTES: *****)

Option Number -

The program stays in prompt mode until a valid option number (1-6) is typed. Typing <CR> causes the program to exit.

Cluster/Scatter Cut-Off Value -

If this value is negative, the program stays in prompt mode.

One-Space Bin Factor -

If this value is negative, the program stays in prompt mode.

Instrumentation threshold

If this value is negative, the program stays in prompt mode.

CDEFAULT (continued)

Note: If the user changes the prompt mode while running CDEFAULT, CDEFAULT will still put out prompts according to the prompt mode that was in effect when CDEFAULT began execution.

*****)

Session with LONG prompts

CURRENT STATUS OF CM FILE PARAMETERS:
CLUSTER/SCATTER CUTOFF VALUE = 500
ONE-SPACE BIN FACTOR = 5
PROMPT FLAG = LONG
INSTRUMENTATION FLAG = ON
INSTRUMENTATION THRESHOLD = 5

PROGRAM OPTIONS:

CHANGE:

1. CUTOFF
2. BIN FACTOR
3. PROMPT FLAG
4. INSTRUMENTATION FLAG
5. INSTRUMENTATION THRESHOLD

OR:

6. SET DEFAULT VALUES FOR 1,2,3,4, AND 5 ABOVE
(500,5,SHORT,OFF,5)

OPTION =/4/

CURRENT STATUS OF CM FILE PARAMETERS:
CLUSTER/SCATTER CUTOFF VALUE = 500
ONE-SPACE BIN FACTOR = 5
PROMPT FLAG = LONG
INSTRUMENTATION FLAG = OFF
INSTRUMENTATION THRESHOLD = 5

PROGRAM OPTIONS:

CHANGE:

1. CUTOFF
2. BIN FACTOR
3. PROMPT FLAG
4. INSTRUMENTATION FLAG
5. INSTRUMENTATION THRESHOLD

OR:

6. SET DEFAULT VALUES FOR 1,2,3,4, AND 5 ABOVE
(500,5,SHORT,OFF,5)

OPTION =/ <CR> /

COMMAND NAME: CDISPLAY

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

If the current display is a two-space scatter plot, CLISPLAY changes the display to a two-space cluster plot, and vice versa. If the current display is a one-space macro plot, CLISPLAY changes the display to a one-space micro plot, and vice versa.

USER INTERACTION: NONE

(NOTES: *****)

If the display code in the DI file is not cluster or scatter, or macro or micro, the message "THE DISPLAY FILE DOES NOT CONTAIN A ONE- OR TWO-SPACE DISPLAY" is printed at the user's terminal.

*****)

COMNOD

COMMAND NAME: COMNOD

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

COMNOD combines two or more lowest nodes which exist on the same level and which share a common node directly above them. Because the common node must be exactly one level above whatever lowest nodes to combine, we can say that nodes to combine must have the same 'parent' node. It is not necessary for all 'children' of the parent to be lowest nodes; children that are not lowest nodes (children that have their own children) may not be combined.

USER INTERACTION:

COMNOD asks you first for the name of the tree which has the nodes you want combined. It then asks you for the name of the parent node of the nodes you want combined. If you select a legal parent, you will see displayed a list of names of children you may combine. This list is labeled 'CLASS SELECT LIST'.

Each name within the list begins with a unique character, the 'class symbol'. You must now enter the class symbols of the nodes you want to combine (see PROMPT NOTES to see three ways of entering class symbols). If your input is in any way not acceptable, COMNOD tells you why and lets you respond again.

COMNOD now asks you for the name of the new node which will be the combination of the nodes you have selected. This name may be the same as any of the nodes combined, or it may be new, whereupon it will be checked for uniqueness among the entire tree structure. If the new name and class symbol are not unique, you will be told why and asked to enter another new name.

Once COMNOD finds the new name acceptable, you will be asked if you want resequencing of the vector id's in the new node. In general, resequencing would be desirable in order to prevent misidentification of vectors in the new class. If you say yes, you will be asked for the identification number for the first vector of the new node. Id's are automatically incremented by one for each succeeding vector of the new node.

EXAMPLE(S):

Session with SHORT prompts

TREE NAME - /SPCTRM/

PARENT NODE NAME - /COLR/

CLASS SELECT LIST

red orng yelo gren blue ingo viol

CLASS SYMBOLS: /biv/

NEW NAME - /blue/

RESEQUENCE ID'S (Y/N)? /Y/

STARTING VECTOR ID. (NUMBER) - /5000/

(PROMPT NOTES: *****)

You may respond to 'CLASS SYMBOLS' in one of three ways:

- 1). A string of class symbols of the nodes to be combined, e.g. 'biv' means "combine the nodes blue, ingo, and viol" (notice that a class symbol is the first character of a node name).
- 2). A minus sign followed by a string of class symbols of the nodes you don't want combined, e.g. '-royg' means "don't combine red, orng, yelo, and gren, but do combine all the others, i.e. blue, ingo, and viol."
- 3). A star (*) preceded or followed by no other characters means 'combine all nodes specified in the CLASS SELECT LIST', e.g. '*' means the same as 'roygbiv', or "combine red, orng, yelo, gren, blue, ingo, and viol."

It is important to remember that the name you give the new node must meet one of the following criterions:

- 1). It may be the name of a node to combine.
- 2). If not the name of a node to combine, it must be a legal OLPARS name not already in use by any node of the current tree, and it must have a class symbol unique among all other lowest nodes of the current tree.

*****)

COMNOD (continued)

Session with LONG prompts

ENTER THE NAME OF THE TREE WHICH CONTAINS
THE NODES TO BE COMBINED - /SPCTRM/

ENTER THE NAME OF THE NODE WHICH IS THE
PARENT OF THE NODES TO BE COMBINED - /COLR/

ENTER A STRING OF CLASS SYMBOLS WHICH REPRESENT THE
NODES TO BE COMBINED.

USE THE 'CLASS SELECT LIST' AS A GUIDE
FOR CHOOSING THE PROPER CLASS SYMBOLS.

IF YOU ENTER A MINUS SIGN AT THE BEGINNING, THE SYMBOLS
FOLLOWING THE MINUS SIGN REPRESENT THE ONLY NODES
IN THE 'CLASS SELECT LIST' NOT TO BE COMBINED.

A STAR '*' ENTERED ALONE REPRESENTS ALL NODES IN
THE 'CLASS SELECT LIST'.

CLASS SELECT LIST

red orng yelo gren blue ingo viol

CLASS SYMBOLS: /-yogr/

ENTER A NAME (4 CHARACTERS MAXIMUM)
FOR THE NEW COMBINED NODE - /prpl/

DO YOU WANT THE VECTOR IDENTIFICATION NUMBERS
IN THE NEW NODE TO BE RESEQUENCED (Y/N)? /y/

ENTER THE STARTING CLASS IDENTIFICATION NUMBER
FOR THE FIRST VECTOR OF THE NEW NODE - /3690/

COMMAND NAME: CRANDTS

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

CRANDTS creates a random data test set from the current data set by extracting a user specified percentage of vectors. The current data set must be structured such that the current node is the senior node of the data tree with only 'lowest' nodes immediately below it (i.e., no intermediate nodes below the senior node are allowed).

(CRANDTS splits the current data set into two separate data sets, one for designing logic (design set), one for testing the designed logic against (the test set). The original data set is essentially destroyed.)

USER INTERACTION:

The user first enters a "new" tree name for the test set being created. Following this, the user will be asked to enter a number representing the percentage of vectors to be extracted from the current data set. Then the user will be prompted for a name to rename the current data set.

EXAMPLE(S):

In the following examples, GRAINS, GRAINTS, AND GRAINDS are all data trees. GRAINS is the current data set, with GRAINTS being the test set, and GRAINDS the current data set after the completion of CRANDTS.

Session with SHORT prompts

TREENAME FOR NEW TEST SET? /GRAINTS/
PERCENTAGE TO EXTRACT - /50/
NEW NAME FOR THE CURRENT DATA SET? /GRAINDS/

CRANDTS (continued)

(PROMPT NOTES: *****)

The current data set name is not allowed to be entered as the test set name (e.g. in the previous example, GRAINS would not be allowed at the first prompt). If the current data set name is entered at the 'NEW NAME FOR CURRENT DATA SET' prompt, the current data set name will remain the same, but the contents of the file will be different.

**** WARNING ****

If CRANDTS fails for some reason after the prompts have all been answered, the current data set will be corrupt.

*****)

Session with LONG prompts

ENTER A NAME FOR THE NEW TREE THAT YOU ARE CREATING.
(8 CHARS. MAX) - /GRAINS/

THE TEST SET NAME IS NOT ALLOWED
TO BE THE CURRENT DATA SET NAME.

ENTER A NAME FOR THE NEW TREE THAT YOU ARE CREATING.
(8 CHARS. MAX) - /GRAINTS/

ENTER A PERCENTAGE THAT WILL REPRESENT THE AMOUNT
OF VECTORS TO BE EXTRACTED FROM THE CURRENT TREE
TO THE NEW TREE. NUMBER ENTERED IS TO BE
BETWEEN 1 - 99 (E.G. 65 OR 62.5) - /50/

ENTER A NAME TO BE USED TO RENAME THE CURRENT DATA SET.
THE CURRENT DATA SET NAME WILL BE CHANGED ONCE THE TEST
SET IS CREATED.
TREENAME? /GRAINDS/

CCOMMAND NAME: CREATLOG

CATEGORY: LOGIC DESIGN CCOMMAND (subsidiary)

FUNCTIONAL DESCRIPTION:

CREATLOG creates a one or two space group logic node in an OLPARS logic tree. After the logic is created, the vectors of the classes present at the logic node will be evaluated against the logic (i.e., to which new logic nodes do the vectors belong?).

This command can only be used after one of the one- or two-space group logic design commands have been called to create a data projection, and a data partition (boundary) has been drawn on the projection.

USER INTERACTION:

User is asked (1) to specify which data classes lie in what regions, (2) whether or not a misclassification error listing is to be sent to the local line printer, and (3) if (s)he is satisfied with the results of the decision logic.

EXAMPLE(S):

A user has projected the current data set using L2EIGV (logic design, two space, eigen vector projection command). A total of seven classes occur in the data set. The user would like to break down future evaluation steps into fewer classes, say two groups of four classes each. The user has drawn a single boundary on the projected data set separating classes 'soy', 'rye', 'alfa', and 'weat' from 'Clov', 'rye', 'corn', and 'oats'. CREATLOG is the next command used.

Session with SHORT prompts

SUPPLY THE CLASS SYMBOL(S) OF THE CLASS(ES)
PRESENT IN FIRST CONVEX REGION

CLASS SELECT LIST

soy rye alfa weat Clov oats corn

CLASS SYMBOLS: /sraw/

CREATLOG (continued)

SUPPLY THE CLASS SYMBOL(S) OF THE CLASS(ES)
PRESENT IN REMAINING REGION

CLASS SELECT LIST

soy rye alfa weat Clov oats corn

CLASS SYMBOLS: /Crco/

<At this point a between-group confusion matrix is
displayed to the user's terminal>

LISTING OF MISCLASSIFIED VECTORS (Y/N)? /Y/
RESULTS OK (Y/N)? /N/
... LOGIC NODE REMAINS INCOMPLETE

(PROMPT NOTES: *****)

The misclassified vectors prompt will not appear if
there are no misclassified vectors.

If the logic created does not produce good results,
an answer of 'NO' to the 'RESULTS OK?' prompt will
tell CREATLOG to leave the logic node incomplete.

*****)

Session with LONG prompts

Do to the length of the prompts, there is no example
given.

COMMAND NAME: CSCALE

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

For one-space displays, when the count option is in effect, CSCALE changes the display to feature the probabilities option, and vice versa. For two-space displays, when rectangular scaling is in effect, CSCALE changes the display to feature square scaling, and vice versa.

USER INTERACTION:

For one-space displays, if the user is in zoom mode and square scaling is in effect, the user is given a message that the option to change from square to rectangular scaling in zoom mode is not available. The user is then asked if he wishes to continue and should type in 'Y' for yes or 'N' for no. If the user chooses to continue, a display featuring square scaling is shown. Otherwise, only the menu is displayed. No other user interaction occurs.

DBNDY

COMMAND NAME: DBNDY

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DBNDY deletes all existing display boundaries from the display file. The projection is then redisplayed with the menu.

USER INTERACTION: NONE

(NOTES: *****)

A message is printed at the users terminal if the projection is not a one- or two-space display.

*****)

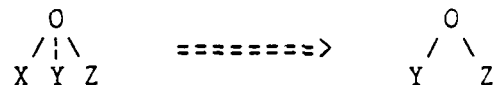
COMMAND NAME: DDATANOD

CATEGORY: UTILITY COMMAND

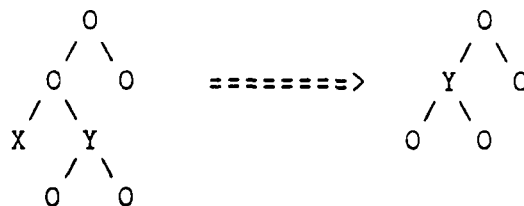
FUNCTIONAL DESCRIPTION:

DDATANOD deletes one or more lowest nodes from a data set. To delete a tree with only one lowest node, DDATATREE must be used. (Note, the 'current' node can not be a lowest node). When the node to be deleted has at least two siblings, the legality of the tree structure is not affected by the deletion of the node. However, if the deletion of the node leaves only one child at the parent, the structure of the tree would become illegal. To solve this structural problem, DDATANOD replaces the parent node with the remaining node. (The parent node name is discarded unless the parent node is the senior node). For example:

With 2 siblings, where X is the node to be deleted -



With 1 sibling, where X is the node to be deleted -



USER INTERACTION:

The user is asked for the class symbol of the node (in the current tree) to be deleted.

DDATANOD (continued)

EXAMPLE(S):

The following examples assume that the tree structure of the current data tree is as follows:



Where Y is the node to be deleted.

Session with SHORT prompts

SELECT ONE NODE FROM THE FOLLOWING LIST TO BE DELETED.

CLASS SELECT LIST

X123 Y123 EFGH

CLASS SYMBOLS: /Y/

SELECT ONE NODE FROM THE FOLLOWING LIST TO BE DELETED.

CLASS SELECT LIST

X123 EFGH

CLASS SYMBOLS: /<CR>/

(PROMPT NOTES: *****)

If the current data tree is a one-class tree, the command 'DDATATREE' must be used to delete the node.

Class Symbols -

If the class symbol typed is invalid, ie., it doesn't exist, DDATANOD will prompt the user again for a valid class symbol. Only one class symbol will be accepted, hence forcing only one node to be deleted at a time.

*****)

Session with LONG prompts

SELECT ONE NODE TO BE DELETED -

CLASS SELECT LIST

X123 Y123 EFGH

CLASS SYMBOLS: /Y/

SELECT ONE NODE TO BE DELETED -

CLASS SELECT LIST

X123 EFGH

CLASS SYMBOLS: /<CR>/

DDATATREE

COMMAND NAME: DDATATREE

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DDATATREE deletes a datatree from the user's directory.

USER INTERACTION:

The user is prompted for a tree name.

EXAMPLE(S):

In the following examples, TESTTREE is the name of the data tree to be deleted.

Session with SHORT prompts

TREENAME? /TESTTREE/
TREENAME? /<CR>/

(PROMPT NOTES: *****)

If the tree name is an invalid name or the tree name does not exist in the tree list, then an error message is displayed and the user will be prompted for another tree name. DDATATREE continues to prompt the user for a tree name until a carriage return (<CR>) is typed.

*****)

Session with LONG prompts

TYPE IN THE OLPARS DATA TREE THAT IS TO
BE DELETED (8 CHARS. MAX.) - /TESTTREE/
TYPE IN THE OLPARS DATA TREE THAT IS TO
BE DELETED (8 CHARS. MAX.) - /<CR>/

COMMAND NAME: DDSUBSTR

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DDSUBSTR permits a user to delete the substructure of an intermediate node in an CLPARS data tree. All the vectors under the intermediate node are merged into that node, which now becomes a lowest node.

USER INTERACTION:

The user is prompted for the name of the tree which contains the substructure to be deleted. Then a prompt appears for the name of the intermediate node above the substructure. If the node is the senior node, the user is requested to confirm the substructure deletion.

When asked for the treename, the user can enter a colon - star ':*'. The colon represents the senior node, the star the current data set. With this combination, the entire data tree structure could be deleted if the user confirms that they want to start at the senior node. The colon can be used with the treename itself if wanted.

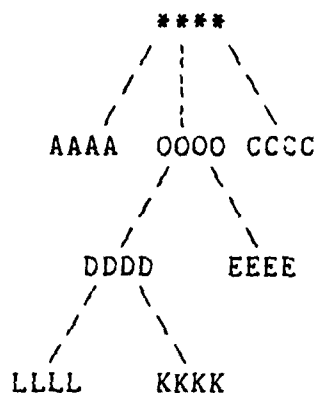
If the node name or class symbol is not unique to the lowest nodes outside the substructure, the user is asked for a new node name. The user is then asked if the vectors of the deleted substructure should have their identifiers resequenced. If the reply is yes, a request for the starting identifier number is given.

If the substructure has been successfully deleted, the user will be informed of this and asked for another tree name, in case there are more substructures to be deleted. Of course, the user may quit at anytime.

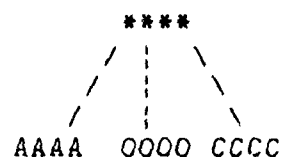
DDSUBSTR (continued)

EXAMPLE(S):

The following examples assume the user has an OLPARS data tree called BIGTREE as the current data set (see figure). The substructure being deleted is below node OOOO.



BEFORE



AFTER

Session with SHORT prompts

TREE NAME? /BIGTREE/
NODE NAME? /****/
CONFIRM (Y/N)? /N/
NODE NAME? /OOOO/
RESEQUENCE VECTORS (Y/N)? /N/

SUBSTRUCTURE HAS BEEN DELETED.

TREE NAME? /<CR>/
<PROGRAM PUTS UP MENU>

(PRCMPT NOTES: *****)

In the given example, the user initially requested to delete the substructure of the senior node, but decided not to do so. The node name prompt reappeared and node 0000 was chosen. At this point the user could be notified that:

- 1) the node does not exist in the tree, in which case a prompt for another node name would appear, or:
- 2) the node has no substructure, that is the node is already a lowest node, in which case the prompt would reappear, or:
- 3) the node name or class symbol may not be unique. In this case the user is prompted for alternate node name to assign to the node in the tree.

Since '0000' does not fall into any of these categories, the resequencing strategy prompt occurred next.

*****)

Session with LONG prompts

ENTER THE NAME OF THE TREE WHICH CONTAINS
THE SUBSTRUCTURE TO BE DELETED - /**/

DO YOU REALLY WISH TO DELETE THE ENTIRE
SUBSTRUCTURE (Y/N)? /N/

ENTER THE NODE NAME OF THE PARENT OF THE
SUBSTRUCTURE TO BE DELETED - /0000/

DO YOU WANT VECTORS RESEQUENCED IN NEW NODE (Y/N)? /Y/

ENTER THE STARTING CLASS IDENTIFICATION NUMBER
FOR THE FIRST VECTOR OF THE NEW NODE - /1/

THE SUBSTRUCTURE HAS BEEN DELETED

ENTER THE NAME OF THE TREE WHICH CONTAINS
THE SUBSTRUCTURE TO BE DELETED - /<CR>/
<THE PROGRAM PUTS UP THE MENU>

DLOGTREE

COMMAND NAME: DLOGTREE

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DLOGTREE deletes a logic tree from the user's directory.

USER INTERACTION:

The user is prompted for a logic tree name.

EXAMPLE(S):

In the following examples, TESTTREE is the name of the logic tree to be deleted.

Session with SHORT prompts

```
TREENAME? /TESTTREE/  
TREENAME? /<CR>/
```

(PROMPT NOTES: *****)

If the tree name is an invalid name or the tree name does not exist in the tree list, then an error message is displayed and the user will be prompted for another tree name. DLOGTREE continues to prompt the user for a tree name until a carriage return (<CR>) is typed.

*****)

Session with LONG prompts

```
TYPE IN THE OLPARS LOGIC TREE THAT IS TO  
BE DELETED (8 CHARS. MAX.) - /TESTREE/  
TYPE IN THE OLPARS LOGIC TREE THAT IS TO  
BE DELETED (8 CHARS. MAX.) - /<CR>/
```

COMMAND NAME: DLSUBSTR

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DLSUBSTR deletes the logic at a specified node of a logic tree.

USER INTERACTION:

The user is prompted for a logic tree name and logic node number from which logic is to be deleted. The user is then asked if (s)he wants to delete: (1) all logic at and below the logic node specified, (2) independent reject strategy, or (3) decision logic only (excluding independent reject strategy). If the logic node selected is the senior logic node, and the user specifies that (s)he wants to delete all logic at that node, the user is asked to confirm the deletion.

If the logic at the user-specified node has successfully been deleted, the user will be informed of this and asked for another logic node number.

DLSUBSTR (continued)

EXAMPLE(S):

In the following example, the name of the logic tree in which a logic substructure is to be deleted is called LOGTREE and the node that is finally deleted is node number 5.

Session with SHORT prompts

LOGIC TREE NAME? /LOGTREE/
LOGIC NODE NUMBER? /1/

DELETE:

- (1) ALL LOGIC
- (2) IND. REJ. STRAT. ONLY
- (3) DECISION LOGIC ONLY

OPTION - /1/

CONFIRM (Y/N)? /N/

NODE NUMBER? /5/

DELETE:

- (1) ALL LOGIC
- (2) IND. REJ. STRAT. ONLY
- (3) DECISION LOGIC ONLY

OPTION - /1/

LOGIC SUBSTRUCTURE HAS SUCCESSFULLY BEEN DELETED.

NODE NUMBER? /<CR>/

(PROMPT NOTES: *****)

In the example above, the user initially requested to delete all logic at the senior node, but decided not to do so. The user was reprompted for the logic node number, and node 5 was chosen. The user again chose option 1, indicating that all logic at and below node 5 should be deleted.

When the senior logic node is specified by the user, and the user chooses option 1, delete all logic, a prompt occurs to make sure the user really wants to delete all logic in the logic tree. If the response is yes, then the deletion occurs as normal. If the response is no, the user is reprompted for the node number, and for the option indicating which type of logic should be deleted at the given node.

If the design data set has not been recently evaluated over the user's logic, DLSUBSTR will print the message 'INCONSISTENCY BETWEEN THE SPECIFIED LOGIC NAME AND THE DESIGN DATA SET'S LOGIC NAME' and then terminate.

Logic Tree Name - When asked for the logic tree name, the user may enter a colon - star ':*'. The colon represents the senior logic node, the star the current logic tree. If the user types the star (*) alone in response to the 'LOGIC TREE NAME' prompt, (s)he will then receive the 'LOGIC NODE NUMBER' prompt. If the user types a colon (:) followed by a treename (may or may not be the current logic tree), the senior node of the tree specified will be referenced.

Logic Node Number - If the user specified node is not in the logic tree, the following message is printed: 'NODE -- IS NOT IN THE TREE', where -- is the specified node number. The user is then reprompted for a valid node number.

*****)

In the following example, LOGTREE is the user's current logic.

Session with LONG prompts

ENTER THE NAME OF THE LOGIC TREE FROM WHICH A
SUBSTRUCTURE IS TO BE DELETED ('*' REPRESENTS
CURRENT LOGIC, INITIAL ':' MEANS START AT SENIOR
NODE) - /*/

DELETE:

(1) ALL LOGIC AT (AND BELOW) LOGIC NODE
(2) INDEPENDENT REJECT STRATEGY ONLY
(3) DECISION LOGIC ONLY (EXCLUDES INDEP. REJ. STRATEGY)
OPTION - /1/

DO YOU REALLY WISH TO DELETE THE ENTIRE LOGIC
TREE STRUCTURE (Y/N)? /N/

ENTER THE NUMBER OF THE LOGIC NODE AT WHICH THE
SUBSTRUCTURE IS TO BE DELETED - /5/

DELETE:

(1) ALL LOGIC AT (AND BELOW) LOGIC NODE
(2) INDEPENDENT REJECT STRATEGY ONLY
(3) DECISION LOGIC ONLY (EXCLUDES INDEP. REJ. STRATEGY)
OPTION - /1/

LOGIC SUBSTRUCTURE HAS SUCCESSFULLY BEEN DELETED.

ENTER THE NUMBER OF THE LOGIC NODE AT WHICH THE
SUBSTRUCTURE IS TO BE DELETED - /<CR>/

DRAWBNDY

COMMAND NAME: DRAWBNDY

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DRAWBNDY lets an CLPARS user draw a partition on a 1-space or 2-space data projection.

For 1-space projections, one or two partitions (thresholds) may be created.

For 2-space projections, one or two partitions (boundaries) may be created. Each boundary may contain up to five line segments.

USER INTERACTION:

If the CLPARS user is in 'short prompt' mode, upon initiation of the DRAWBNDY command, a graphics cursor will be displayed. If 'long prompt' mode is active, the display is erased and an explanation for drawing a data partition is given.

FOR 1-SPACE PROJECTIONS ONLY:

Position the graphics cursor on the spot where the threshold is desired.

Type any alphanumeric key on the terminal (except 'Q') to enter the threshold value into the current dataset. A vertical line will be drawn on the display screen at the threshold point.

Repeat the above steps to enter a second threshold. Drawbndy will terminate after the second threshold has been entered.

If the 'Q' key is typed in either instance, the threshold which would have been created is ignored, and DRAWBNDY terminates.

FOR 2-SPACE PROJECTIONS ONLY:

You are notified that you are in 'BCUNDARY entry mode' by an intensified 'BCUNDARY:' at the lower left hand corner of the display.

Position the graphics cursor anywhere on the display screen in order to create the starting point of the first boundary.

Type any alphanumeric key on the display terminal (except 'Q') to enter the starting point (DRAWBNDY will 'remember' the starting point while you move the graphics cursor to the second point).

If you do type 'Q', DRAWBNDY will 'quit' (terminate) because the starting point has not been entered yet.

Move the graphics cursor to another position, which will be the second point of the boundary. If you enter the point by typing an alphanumeric key (besides 'Q'), you will see a line segment drawn from the starting point to the second point.

If instead of entering the second point you type 'Q', you will have an incomplete boundary. Should this occur, DRAWBNDY will display an error message and terminate.

You may consider the one line segment as the boundary, or you can keep adding line segments to the boundary in a similar manner. Each new line segment will be drawn from the previous point entered to the new point.

DRAWBNDY will let you enter up to six points (five line segments) for each boundary. If you wish to enter fewer than six, you can quit by typing 'Q' after you have drawn the last line of the boundary.

After you have drawn either five line segments or typed 'Q', DRAWBNDY will display 'CONVX PT:' beneath 'BCUNDARY:'. When you see this, you must position the graphics cursor somewhere within the convex region of the boundary (the side which would be within a convex polygon if the boundary were to be closed).

Enter the convex point by typing any alphanumeric key but 'Q'. DRAWBNDY will now enter information on the new boundary into the current dataset.

DRAWBNDY (continued)

If instead of entering the convex point, you enter 'Q', DRAWBNDY will display an error message and terminate. Whenever the convex point is 'missing' or the boundary is 'incomplete', as stated in an error message, no information on that boundary will be stored in the current dataset.

After you have entered the convex point of the first boundary, DRAWBNDY will let you draw a second boundary. Again, you are notified you are in 'BOUNDARY entry mode' by an intensified 'BOUNDARY: '.

If you want to stop at this point (one boundary was enough) press the 'Q' key. DRAWBNDY will terminate.

Otherwise, position the cursor to the starting point of the second boundary, enter it by hitting any key but 'Q', and proceed as before.

After you enter the convex point for your second boundary, DRAWBNDY will terminate.

EXAMPLE(S):

In the examples which follow please note the meaning of two symbols:

... means 'position the graphics cursor'
K means 'any alphanumeric key
except 'Q''

The current entry mode (BOUNDARY, CONVEX PT., THRESHOLD) is shown to the left of the user input.

Sessions with SHORT prompts

FOR 1-SPACE PROJECTIONS ONLY:

To create two thresholds:

THRESHOLD: /...K/
THRESHOLD: /...K/

To create one threshold:

THRESHOLD: /...K/
THRESHOLD: /Q/

FOR 2-SPACE PROJECTIONS ONLY:

To create two boundaries - the first consisting of three line segments, the second consisting of five:

```
BCUNDARY:  /...K...K...K...KQ/
CCNVX PT:  /...K/
ECUNDARY:  /...K...K...K...K...K...K/
CCNVX PT:  /...K/
```

To create one boundary consisting of only one line segment:

```
BCUNDARY:  /...K...KQ/
CCNVX PT:  /...K/
ECUNDARY:  /Q/
```

(*****)

Session with LONG prompts

Due to the limitation of visible space on the display screen there is no interactive 'SHORT' and 'LONG' prompts (i.e., the CLPARS program help function is not available in a graphics input program).

However, if you have changed the default prompt mode to be 'LONG' by means of the 'CDEFAULT' command, you will receive a preliminary explanation of how to respond to the prompts issued by DRAWBNDY.

DRAWLOG

COMMAND NAME: DRAWLOG

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DRAWLOG produces a pictorial display of a portion (or all) of a logic tree at any stage in the development of the logic. Logic nodes are displayed as partitioned boxes with interconnecting lines to illustrate their relationship to each other.

EXAMPLE LOGIC NODE

```
-----
      ----independent reject strategy indicator
      |  --logic node number
      |  |
      |  v  v
      |-----
      | * 2 | NMV | <--- logic type
      |-----
      | abCsoW | <--- class(es) present at node
      |-----
```

The class(es) present at a logic node are listed in the bottom portion of the box of every logic node (except REJECT nodes). If the node is a completed logic node, there is only one class present, and its name appears in this portion of the box. Otherwise the class display symbols of the classes present at the node appear in this region.

In the upper left portion of every logic node box is the logic node number preceded by either a star or a blank, depending upon whether an independent Boolean reject strategy is associated with the node or not.

The upper right portion of the logic node box contains the type of logic at the node (e.g. NMV), 'INCMPLT' for incomplete logic nodes, or '*REJECT*' for reject nodes.

If a given logic tree is too large to display on the screen, only a portion of the tree is displayed; with the user option of displaying more.

USER INTERACTION:

The user is asked for the name of a logic tree, and a logic node number to determine the structure to be displayed.

EXAMPLE(S):

In the following example, the logic tree being drawn is called GRAINS. The user has requested to see the entire tree.

Session with SHORT prompts

LOGIC TREE NAME? /GRAINS/
LOGIC NODE NUMBER? /1/

(the screen is erased and the logic tree is displayed)

(PROMPT NOTES: *****)

There is a short hand notation for obtaining a picture of the "current" logic tree. Instead of typing the name of the tree at the 'LOGIC TREE NAME?' prompt, type an asterisk (*); then the current logic tree name will be used by DRAWLOG (the logic node number prompt will follow). If a leading colon (:) is typed with the logic tree name (e.g. :GRAINS), the senior node (node number 1) will be used as the starting node, and no 'LOGIC NODE NUMBER?' prompt will appear. Thus, if ':*' is typed in at the 'LOGIC TREE NAME?' prompt, the senior node of the current logic tree will be the starting node at which DRAWLOG will display the tree.

PAGING ***> If a particular level in the logic tree is only partially displayed, DRAWLOG informs the user of this fact by placing a zero (0) at the top of the display, and/or a minus one (-1) at the bottom of the display. The 'LOGIC NODE NUMBER?' prompt will appear at the bottom of the display. If zero appears as a logic node number on the display, then a zero may be entered at the prompt. This will cause the next "page" of nodes to be displayed. If minus one appears as a logic node number on the display, and is entered at the prompt, then the previous "page" of nodes is displayed.

DRAWLOG (continued)

During "paging" mode, the user may also enter a "negative" node number (e.g. -21) at the prompt. The requested node will appear as the first node (at the bottom of the display) of a "paged" logic tree display. This option allows a user direct viewing to a specific portion of the tree.

When there are more logic tree levels than can be shown at one time, extra lines are drawn radiating out from the right side of each logic node box which has an undisplayed structure beneath it. The 'LOGIC NODE NUMBER?' prompt will appear so that the user has the ability to see the undisplayed portion of the tree (i.e., the user enters the number of the node showing radiating lines).

Once some sort of "paging" occurs, DRAWLOG remains active until the user tells it to exit via the OLPARS program exit convention.

*****)

In the following example, GRAINS is the current logic tree and the entire tree is to be shown.

Session with LONG prompts

ENTER THE NAME OF THE LOGIC TREE TO BE
DRAWN ('*' REPRESENTS CURRENT LOGIC,
INITIAL ':' MEANS START AT SENIOR NODE) - /*/*

(the screen is erased and the logic tree is displayed)

COMMAND NAME: DRAWTREE

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DRAWTREE displays a selected OLPARS data tree, showing the structural relationship between the various nodes of the tree.

USER INTERACTION:

The user is asked for the tree name and node name of the data set to be drawn.

EXAMPLE(S):

In the following example, the data tree being drawn is called GRAINS. The user has requested to see the entire tree.

Session with SHORT prompts

TREE NAME? /GRAINS/
NODE NAME? /****/

(the screen is erased and the data tree is displayed)

(PROMPT NOTES: *****)

In the short prompt example, the user requested the entire tree to be drawn, because the senior node '****' was given at the node name prompt. If the tree or node name did not exist, the user would have been notified.

If the current data tree is to be drawn, the user can specify this using a short hand notation. At the 'TREE NAME?' prompt, type in an asterisk (*); then the current data tree name will be used by DRAWTREE (the node name prompt will follow). If a colon (:) precedes the data tree name (e.g. :GRAINS), the tree is displayed starting at the senior node (no node prompt will appear when a colon is used). Thus, if ':*' is typed in at the 'TREE NAME?' prompt, the senior node of the current data tree will be displayed, starting at the senior node (see next example).

DRAWTREE (continued)

PAGING ***> If a particular level in the data tree is only partially displayed, DRAWTREE informs the user of this fact by placing a page number in the upper left corner of the display. The 'NODE NAME?' prompt will appear at the bottom of the display. At this point the user can type a comma (,) to advance to the next "page" of the display. To return to the previous "page", the user types two commas (,,) to the 'NODE NAME?' prompt.

When there are more data tree levels than can be shown at one time, extra lines are drawn radiating out from the right side of each data node which has an undisplayed structure beneath it. The 'NODE NAME?' prompt will appear so that the user has the ability to see the undisplayed portion of the tree (i.e., the user enters the name of a node showing radiating lines).

Once some sort of "paging" occurs, DRAWTREE remains active until the user tells it to exit via the CLPARS program exit convention.

*****)

In the following example, the user requested to draw the current data tree, starting at the senior node.

Session with LONG prompts

ENTER THE NAME OF THE TREE
WHICH YOU WANT TO DRAW - /:*/

(the screen is erased and the data tree is displayed)

COMMAND NAME: DSCRMEAS

CATEGORY: MEASUREMENT EVALUATION COMMAND

FUNCTIONAL DESCRIPTION:

DSCRMEAS computes the discriminant measurement evaluation statistics and outputs to the screen an overall ranking of the measurements for the current data set. An overall ranking consists of a list of measurements, the corresponding discriminant measurement values (in descending order), a list of class symbols and a list of class pair symbols, both of which correspond to the measurement number. The discriminant measurement values displayed are those values which separate all classes on the basis of a given measurement. The measurement numbers which correspond to the largest discriminant values (those at the top of the list) are in general ("overall") the best measurements. The class symbol displayed represents the class which is best separated from all other classes using the given measurement ("best class for measurement x"). The class pair symbol represents the two classes which are best separated from each other using the given measurement ("best class pair for measurement x").

Note, DSCRMEAS can be used in excess measurement mode.

USER INTERACTION:

The user is asked to select one of the following options for computing the discriminant measurement evaluation statistics:

- (1) weight classes equally
- (2) weight classes by the number of vectors in each class

The option number selected by the user is used by DSCRMEAS to determine the number (weighting factor) by which to multiply the variance of a given class. If the user selects Option 1, the variance of a class is multiplied by a number which equals the total number of vectors in the data set divided by the number of classes in the data set. If the user selects Option 2, the variance of a class is multiplied by the number of vectors in the given class.

There will be more user interaction only if an exception condition (zero denominator) occurs in the discriminant measurement calculation. This will occur if the variance of any measurement is zero for more than one data class,

DSCRMEAS (continued)

or if more than one data class contains only one vector. If an exception condition occurs, the user is asked if a listing of the exception conditions should be printed. If the user indicates no listing is to be produced, (s)he is then asked if command execution should continue. If the user indicates a listing is to be produced, exception conditions are printed at the terminal. If there is more than one page of output, the user is prompted for the view of the next page. If an answer of 'Y(es)' is given, exception conditions continue to be printed. If all output fits on one page, or if the user does not want to see the next page, a prompt for command continuation is given. Thus, the user has the opportunity to terminate output of exception conditions or terminate execution of the command after a page of output has been printed.

EXAMPLE(S):

Session with SHORT prompts

In the following example, the data set contains four classes and the vector dimensionality is four. DSCRMEAS results show that measurement 4 is the "overall" best measurement. Also, the class best separated from all other classes by measurement 4 ("best class for measurement 4") is class C, and the two classes best separated from each other by measurement 4 ("best class pair for measurement 4") are classes A and C.

SELECT AN OPTION:

(1) WEIGHT CLASSES EQUALLY

(2) WEIGHT CLASSES BY THE NUMBER OF VECTORS IN EACH CLASS

OPTION # = /1/

A RANK ORDER DISPLAY

DATE: 13-MAR-81 02:16:36

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| 4 | 8.3926E+00 | C | A/C |
| 2 | 8.3155E+00 | A | A/C |
| 1 | 7.5638E+00 | C | B/C |
| 3 | 6.9224E+00 | B | A/D |

(PROMPT NOTES: *****)

DSCRMEAS will remain in prompt mode until the user types a '1' or a '2' or the OLPARS character for exiting commands. A 'class pair symbol' is defined to be 2 class symbols separated by a slash (/).

*****)

Session with LONG prompts

The following example assumes the variance for measurement 2 was equal to zero in both class A and class D.

TYPE A '1' OR A '2' DEPENDING ON THE DESIRED OPTION #
 (1) WEIGHT CLASSES EQUALLY
 (2) WEIGHT CLASSES BY THE NUMBER OF VECTORS IN EACH CLASS
 OPTION NUMBER = /2/

EXCEPTION CONDITION(S) INVOLVING A ZERO DENOMINATOR
 OCCURRED IN THE CLASS PAIR DISCRIMINANT MEASUREMENT
 CALCULATION(S)

DO YOU WANT A LISTING OF THESE EXCEPTIONS (Y/N)? /Y/

EXCEPTIONS TO DSCRMEAS CALCULATIONS. THE DISCRIMINANT
 VALUE FOR EACH PAIR LISTED BELOW WAS SET TO ZERO.

PAGE 1

| MEAS | CLASS/# | VECTORS | CLASS/# | VECTORS | MEAN DIFFER. |
|------|---------|---------|---------|---------|--------------|
| 2 | A | 10 | D | 10 | 0.0000E-01 |

CONTINUE (Y/N)? /N/

DTRENAME

COMMAND NAME: DTRENAME

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

Change the name of an OLPARS data tree.

USER INTERACTION:

The name of the data tree to be changed is requested from the user, along with the new name of the tree.

EXAMPLE(S):

The following example shows the tree 'GRAINS' being renamed to 'CEREALS'.

Session with SHORT prompts

FROM? /GRAINS/
TO? /CEREALS/

(PROMPT NOTES: *****)

If the new name of the tree is already in use, the user is requested for permission to destroy the existing tree.

*****)

In the following example, the tree 'CEREALS' already exists.

Session with LONG prompts

ENTER NAME OF TREE TO BE RENAMED - /GRAIN1/
ENTER NEW NAME OF TREE - /CEREALS/
THE NAME 'CEREALS' IS IN USE IN YOUR DIRECTORY;
DO YOU WANT TO DESTROY THE DATA SET WITH THAT NAME (Y/N)?
/Y/

COMMAND NAME: DVEC

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

DVEC deletes vectors from a class within the current data set. All vectors, a range of vectors, or individual vectors from a specified data class may be deleted.

USER INTERACTION:

The user is requested to pick the class from which vectors are to be deleted. The user then chooses the mode of vector deletion (all vectors, a range of vectors or single vectors).

If a range of vectors is to be deleted, the user is asked for beginning and ending vector identifiers which bound the range of the vectors to be deleted.

If single vectors are to be deleted, the user is asked for the individual identifiers of those vectors.

EXAMPLE(S):

In the following example, the data class from which vectors are going to be deleted is 'oats'. All vectors will be deleted from the class. The data node will also be deleted because no vectors remain in the class.

(All examples assume the current data set is GRAINS-****, with 'soy', 'oats', 'weat', 'rye', 'corn', 'Clov', and 'alfa' as data classes.)

Session with SHORT prompts

CLASS SELECT LIST

soy oats wheat rye corn Clov alfa

CLASS SYMBOLS: /o/

DELETE MODE (A/R/S)? /A/

...TOTAL NUMBER OF VECTORS DELETED = 35

DVEC (continued)

=====

In the next example, a range of vectors from data class 'soy' will be deleted.

CLASS SELECT LIST

soy weat rye corn Clov alfa

CLASS SYMBOLS: /s/
DELETE MODE (A/R/S)? /r/
RANGE (INITIAL, LAST)? /5,20/

...TOTAL NUMBER OF VECTORS DELETED = 4

=====

For the following example, only one vector is going to be deleted from the data class 'weat'.

CLASS SELECT LIST

soy weat rye corn Clov alfa

CLASS SYMBOLS: /w/
DELETE MODE (A/R/S)? /s/
VECTOR ID(S) (END WITH ZERO)? /371 0/

...TOTAL NUMBER OF VECTORS DELETED = 1

(PROMPT NOTES: *****)

In the second and third examples, note that class 'oats' was missing from the class select list because it was deleted in the first example.

Also, in the second example, note that only four vectors were deleted, because only four vectors fell within the specified range.

If a data class becomes empty during range or single vector deletion, the data class node is deleted from the tree and the user is notified.

During single vector deletion, the maximum number of vector ids. that can be entered is twenty.

*****)

Session with LONG prompts

The following example shows what occurs when all the vectors from a data class have been deleted when using single vector deletion.

PICK CLASS FROM WHICH VECTORS ARE TO BE DELETED

CLASS SELECT LIST

soy oats weat rye corn Clov alfa

CLASS SYMBOLS: /r/

DELETE 1) ALL VECTORS

2) RANGE OF VECTORS

3) SINGLE VECTOR(S) -- (A/R/S)? /S/

ENTER NUMERIC IDENTIFICATION NUMBER(S)

(SEPARATED BY SPACES) OF VECTORS TO BE DELETED

(LAST ID. MUST BE ZERO, MAX. 20) - /17 16

35 3000 75 0/

(ALL VECTORS IN CLASS rye HAVE BEEN DELETED.

THE NODE HAS BEEN DELETED FROM THE DATA TREE.)

...TOTAL NUMBER OF VECTORS DELETED = 5

EIGNXFRM

COMMAND NAME: EIGNXFRM

CATEGORY: TRANSFORMATION COMMAND

FUNCTIONAL DESCRIPTION:

EIGNXFRM generates a new tree of equal or lower dimensionality by transforming the current data set, using the eigenvectors which correspond to the selected eigenvalues (the selected eigenvalues consist of the threshold eigenvalue and all eigenvalues above it in the list).

USER INTERACTION:

The user is asked for the following information:

- 1) The name of the tree to be created as a result of the transformation.
- 2) If a lineprinter listing of eigenvalues is desired.
- 3) The number of the threshold eigenvalue. This will be the number of eigenvectors used to create the transformation matrix.
- 4) If the transformation matrix should be saved, and if so, the name of the saved transformation matrix.

EXAMPLE(S):

Session with SHORT prompts

In the following example, the current data set has a dimensionality of six. The four largest eigenvalues are selected, and the corresponding eigenvectors are used to create a transformation matrix. The data set is transformed, and a tree called NEWTREE is created.

NEW TREE NAME? /NEWTREE/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 9.475473E+02 |
| 2 | 1.921622E+02 |
| 3 | 3.763162E+01 |
| 4 | 7.532745E+00 |
| 5 | 3.116945E+00 |
| 6 | 1.605199E+00 |

EIGNXFRM (continued)

PRINTOUT (Y/N)? /N/
NUMBER (POSITION) OF THE THRESHOLD EIGENVALUE: /4/
TRANSFORMATION COMPLETE
SAVE TRANSFORMATION MATRIX (Y/N)? /N/

(PROMPT NOTES: *****)

Tree Name -

If the tree name entered by the user is the same as the current tree name, an error message is printed and 'EIGNXFRM' remains in prompt mode. If the tree name entered already exists, but is not the same as the current tree name, the user is asked if s(he) wants to destroy the existing tree. If the user types 'Y' (for yes), 'EIGNXFRM' creates a new tree with the user-specified name. If the user types 'N' (for no), 'EIGNXFRM' asks for another tree name.

Eigenvalue Threshold -

The user is prompted for an eigenvalue threshold until a number greater than zero and less than or equal to the dimensionality is typed.

Saved Matrix Name -

If the user wants to save the transformation matrix, and there is no available space in the Saved Matrix (SM) File, a message is printed and EIGNXFRM exits. (NOTE: Although the transformation procedure was successful, the user will be unable to save the matrix used in this transformation. Try again later after deleting some saved matrices from the SM File.) The matrix name entered cannot exceed 8 characters of which the first must be alphabetic and the remainder alphanumeric. The matrix name must also be unique within the SM file. The user is prompted until a valid and unique matrix name is entered.

*****)

EIGNXFRM (continued)

Session with LONG prompts

In the following example, the tree NEWTREE already exists in the user's directory, and the user decides to destroy it. The transformation matrix is saved and is given the name EIGEN1.

TYPE IN NAME TO GIVE TO THE OLPARS DATA TREE THAT
WILL BE CREATED AS A RESULT OF THE TRANSFORMATION
/NEWTREE/

THE NAME "newtree" IS IN USE IN YOUR DIRECTORY; DO YOU
WANT TO DESTROY THE DATA SET WITH THAT NAME (Y/N)? /Y/

| NUMBER | EIGENVALUE |
|--------|------------|
| . | . |
| : | : |
| . | . |

DO YOU WANT A PRINTOUT OF THE EIGENVALUES (Y/N)? /N/

THE TRANSFORMATION MATRIX WILL CONSIST OF EIGENVECTORS
WHICH CORRESPOND TO THOSE EIGENVALUES ABOVE AND
INCLUDING THE THRESHOLD EIGENVALUE.

NUMBER (POSITION) OF THE THRESHOLD EIGENVALUE: /4/

TRANSFORMATION COMPLETE

DO YOU WISH TO SAVE THE TRANSFORMATION MATRIX (Y/N)? /Y/

TYPE IN THE NAME OF THE MATRIX TO BE SAVED - /EIGEN1/

COMMAND NAME: FILEIN

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

FILEIN creates an OLPARS data tree from a "system" text file with the following format.

```

----- 1-4 character class name,
         [vector identifier,]
1       [keyword 1,]
         .
v       . (keywords have an arbitrary
e       . length and may be imbedded
c       . with blanks)
t       [keyword N,]
o       X1,
r       .
         . (vector elements)
         .
----- Xndim;
         .
         .
         .
         [/*] (end of file indicator)

```

Note: [] means optional field.

The vectors are separated with semicolons (;). Therefore, there may be more than one vector per input data line, or conversely, there may be more than one input data line per vector.

ALL vectors in the file to be converted to an OLPARS data tree must have the same format.

E.G., - the following vector has a vector id.,
2 keywords, and 4 vector elements.

expl, 1, this is an example vector, dim. of 4,
1,2,3,45.7;

- the following vectors have no vector ids.
or keywords, and there are 5 vector elements
in each.

ex, 1.2,3.0, 2, 5 , 7.9; eg,1,2,3,4,5;

FILEIN will report the first 5 vectors that have any format errors. After that, it will remain quiet

FILEIN (continued)

until the end of processing the input file. A total count of errors encountered will be given. If five or less errors are encountered, FILEIN will still create the data tree. WARNING: the data in this tree may or may not be what you expect it to be.

Note, FILEIN can be used in excess measurement mode.

USER INTERACTION:

The user is asked:

- 1) for a file name - the name of the file to be converted to an OLPARS data tree.
- 2) for a tree name - the name of the OLPARS data tree to be created.
- 3) whether or not there is a numeric vector identifier on each vector.
- 4) for the number of keywords on each vector.
- 5) for the dimensionality of the data set.

EXAMPLE(S):

The following example assumes the user has his/her vectors in the file "FILEIN.DAT" (located in user's OLPAR directory) with a format similar to the one given in the first example above.

Session with SHORT prompts

```
-----  
FILENAME? /FILEIN.DAT/  
TREE NAME? /MYTREE/  
VECTOR ID. (Y/N)? /Y/  
KEYWORD COUNT (NUMBER)? /2/  
VECTOR DIMENSIONALITY? /4/
```

(PROMPT NOTES: *****)

Filename -

When FILEIN can't open the file specified here (because it doesn't exist or for some other obscure "system" dependent reason), it will inform you of the problem and then prompt you for another file name.

Tree name -

If the tree name specified already exists, FILEIN will ask if you want the existing tree destroyed. If you want the tree destroyed, type in a "Y" (for yes). User interaction will then continue as above. If you don't want the tree destroyed, type in a "N" (for no). FILEIN will then ask you for another tree name.

Vector identifier -

Each vector is allowed to have a user assigned numeric identification code (maximum value limited by local integer size). If no vector identifier is present on the input-vector (indicated to FILEIN by typing 'N' to the prompt), FILEIN will assign a unique identifier to each incoming vector.

Keyword count -

If there are no keywords on any of the vectors, type a zero (0).

Vector dimensionality -

If you don't happen to know how many measurements are in each vector of the input data set (shame on you), type in a zero. The first vector in the data set will be used to determine the vector dimensionality.

Note: FILEIN can currently create what can be considered an 'invalid' OLPARS data tree. The tree consists of a senior node and one lowest node (usually there must be at least two lowest nodes under any intermediate node). Throughout OLPARS there is no other way to obtain a tree with only one lowest node under an intermediate node. This specialized tree is being allowed so that single vectors or classes may be added to existing trees via the APPEND command.

*****)

FILEIN (continued)

Session with LONG prompts

In this example, the user has the FILEIN input file "USERDATA" in their OLPARS directory. There are no vector identifiers or keywords on any of the vectors. Each vector has 25 elements.

TYPE IN NAME OF DATA FILE TO BE CONVERTED TO OLPARS
DATA TREE - /USERDATA/

TYPE IN NAME TO GIVE TO THE OLPARS DATA TREE YOU WANT
TO CREATE (8 CHARS. MAX.) - /MYTREE2/

IS THERE A VECTOR IDENTIFIER (NUMBER :
PRESENT ON EACH RECORD (Y/N)? /N/

TYPE IN THE NUMBER OF KEYWORDS PRESENT ON EACH
VECTOR - /0/

HOW MANY MEASUREMENTS ARE IN AN INDIVIDUAL VECTOR
(TYPE 0, IF UNKNOWN)? /25/

COMMAND NAME: FILEOUT

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

FILEOUT places the vectors of an CLPARS data tree into a "system" text file with the following format:

OLPARS data file output format

```

-----> 1-4 character class name,
      x1
      .      (vector elements)
      .
      .
one vector -> xndim;
      .
      .
      /*      (end of file indicator)
  
```

Note, FILEOUT can be used in excess measurement mode.

USER INTERACTION:

The user specifies:

1. the tree to be converted to a "system" file
2. the "system" file name
3. the field width of a vector measurement
4. the number of decimal positions in the measurement output field (precision)

EXAMPLE (S):

The following example shows the OLPARS data tree 'GRAINS' being dumped into the "system" file 'GRAINS.TXT'. The measurements of a vector are to be turned into floating point format with a field width of nine and precision of four (e.g. the number 89.34765 will look like ' 89.3476')

Session with SHORT prompts

```

-----
TREE NAME? /GRAINS/
FILENAME? /GRAINS.TXT/
EXPONENTIAL FORMAT (Y/N)? /N/
FIELD WIDTH (2-15) - /9/
DECIMAL PRECISION (0-7) - /4/
  
```

FILEOUT (continued)

(PROMPT NOTES: *****)

At the tree name prompt, the user may specify the current data set by typing '*'.

When using the floating point format, the total field width specified must be large enough to accommodate a minus sign (plus signs are suppressed), at least one digit to the left of the decimal point, the decimal point itself, and <precision> digits to the right of the decimal. For this reason, <field width> should always be greater than or equal to <precision> plus three.

When using 'scientific' notation format, the total field must accommodate a possible minus sign, space for a single numeric digit, a decimal point, and a four-place exponent (e.g. E-99), in addition to <precision> digits. Therefore, <field width> should always be greater than or equal to <precision> plus seven.

Note, it is possible to obtain an output conversion error from FILEOUT if a measurement is too large for the <field width>-<precision> specification. When this happens, the user is shown the class, vector id. and measurement number where the conversion error has occurred. FILEOUT then halts execution.

*****)

In the next example, the current data set is going to be dumped into the text file 'GRIT'. The data vectors contain large exponents so they will be dumped using exponential format.

Session with LONG prompts

ENTER THE NAME OF THE DATA SET TO BE
USED ('*' REPRESENTS CURRENT DATA SET) - /*/

TYPE IN NAME OF FILE TO CONTAIN DATA VECTORS - /GRIT/

DO YOU WANT THE VECTOR MEASUREMENTS TO BE PLACED
IN SCIENTIFIC NOTATION (<DISABLE FOR NUMBERS THAT
EXCEED FLOATING POINT FORMAT) (Y/N)? /Y/

TYPE IN OUTPUT FIELD WIDTH OF A MEASUREMENT (7-15) - /11/

TYPE IN DECIMAL PRECISION OF A MEASUREMENT (0-4) - /3/

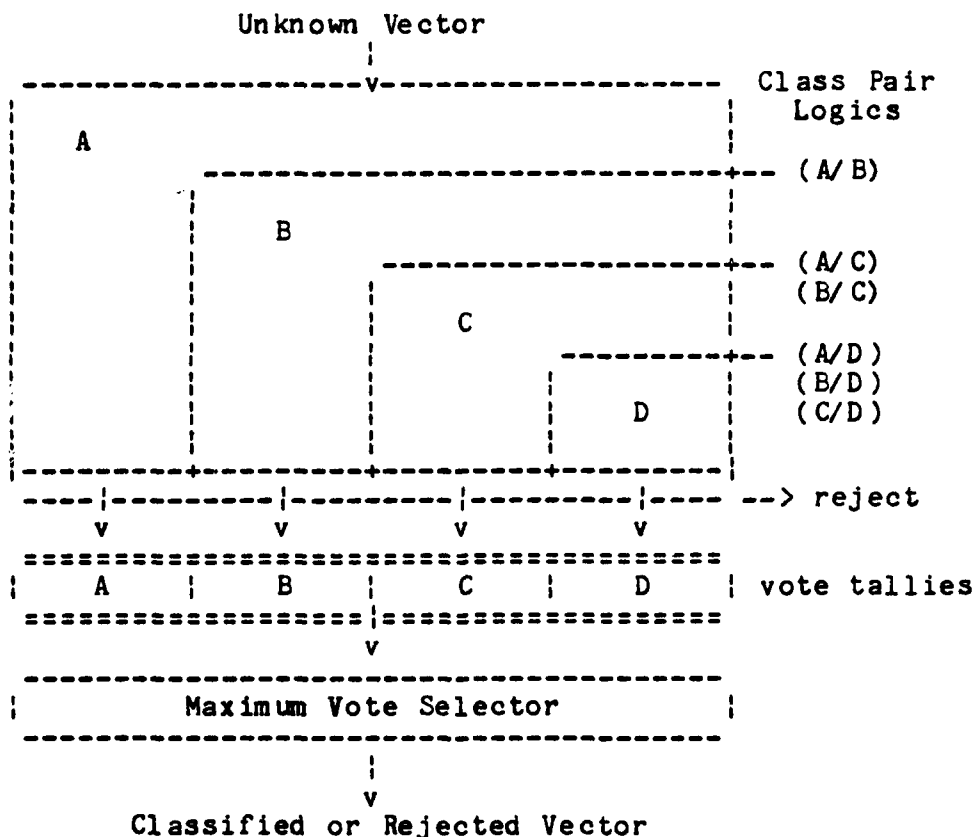
COMMAND NAME: FISHER

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

FISHER initiates pairwise logic for a user specified logic node. At the given logic node, Fisher pairwise logic is created. The logic is constructed by computing optimal linear discriminants and 5 thresholds for every class pair within the group of classes present at the given logic node. During logic evaluation, the discriminants and thresholds are used to distinguish one produces a vote, indicating the class to which the vector belongs. (Note, the vector may be rejected if the discriminant thresholds selected contain reject regions, or the maximum vote count is not adequate, as indicated by the minimum vote threshold.)

The following diagram is a visual aide depicting an unknown vector passing through Fisher logic created for four classes.



FISHER (continued)

Measurements may be ignored during computation of the Fisher discriminant.

To evaluate the newly created logic, use PWEVAL.

Modifications may be made to the Fisher* logic using one of the pairwise modification commands (e.g. FISHMOD).

* Fisher pairwise logic refers to a one-space logic, based on the Fisher direction, for each possible pair of classes present at the node for which logic is being designed.

USER INTERACTION:

The user is asked (1) if all vectors from the classes which lie at the logic node should be used, (2) whether there are measurements to be ignored, (3) for a minimum vote count, (4) for a number of thresholds.

EXAMPLE(S):

In the following example, the user has decided to use only those vectors which lie at the logic node, to ignore the second measurement, to use a minimum vote count of 2, and 4 thresholds.

Session with SHORT prompts

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC NODE? /N/
IGNORE MEASUREMENTS (Y/N)? /Y/
MEASUREMENT # /2/
MEASUREMENT # <CR>
MINIMUM VOTE COUNT? /2/
NUMBER OF THRESHOLDS? /4/

(The program puts up the menu)

(PROMPT NOTES: *****)

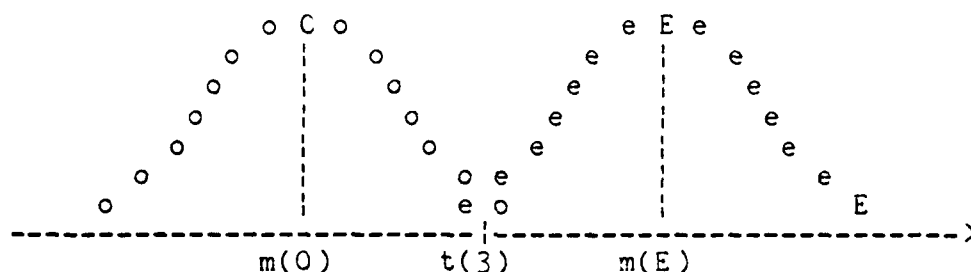
Note, <CR> is entered to complete the ignore measurement cycle. If <CR> is entered at the first measurement prompt, however, the command will exit.

There is an upper limit to the minimum vote count; one less than the number of classes present at the

logic node.

FISHER allows the selection of 1, 2, 3, or 4 thresholds. With one threshold, the vote is either for one class or the other, and the threshold (3) is exactly between the projection of the two class means (see figure a).

a) One Threshold for class pair (E vs C)



Vote for class 'E' if $(D, X) > t(3)$

Vote for class 'O' if $(D, X) \leq t(3)$

=====
Notational Description for figures

$m(O)$ - mean of class 'O'
 $t(1)$ - first threshold (5 thresholds total)
 $d(ij)$ - discriminant vector of the i th and j th class
 D - $d(ij)$
 X - arbitrary class vector
 (D, X) - vector X evaluated against discriminant

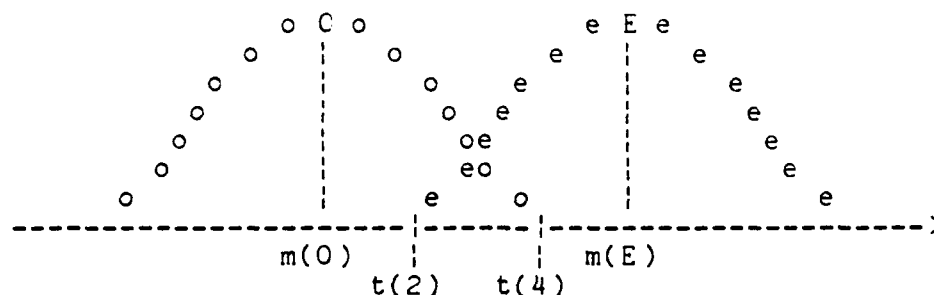
 $>$ - relational operator 'greater than'
 $=>$ - relational operator 'greater than or equal to'
 $<$ - relational operator 'less than'
 $<=$ - relational operator 'less than or equal to'

=====

FISHER (continued)

With two thresholds, the vote can be for either class, or a reject if the projection of the vector falls between threshold 2 and 4 (see figure b).

b) Two Thresholds for class pair (E vs O)



Vote for class 'E' if $(D, X) > t(4)$

Vote for class 'O' if $(D, X) < t(2)$

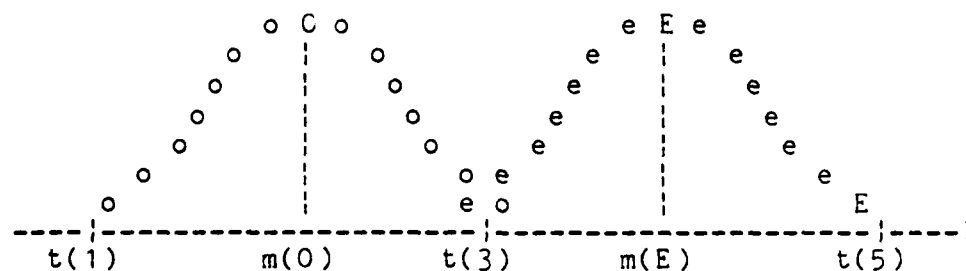
NO VOTE if $t(2) \leq (D, X) \leq t(4)$

=====

=====

With three thresholds, the vote can be rejected if it falls outside the far left and right thresholds, 1 and 5, respectively. (see figure c).

c) Three Thresholds for class pair (E vs O)



Vote for class 'E' if $t(3) < (D, X) < t(5)$

Vote for class 'O' if $t(1) < (D, X) \leq t(3)$

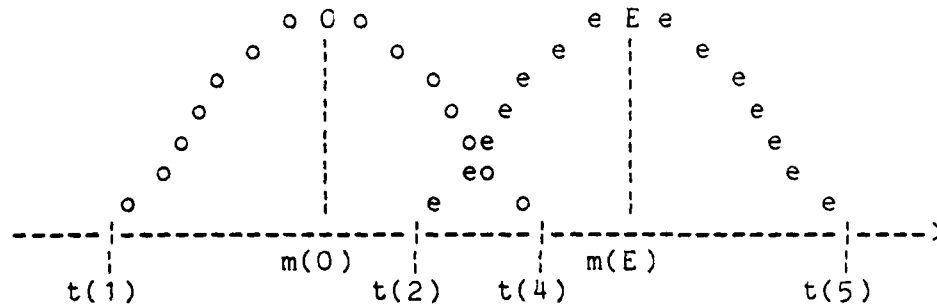
NO VOTE if $(D, X) \leq t(1)$ or $(D, X) \geq t(5)$

=====

=====

With four thresholds (1,2,4, and 5 are in effect), the vector must fall within the neighborhood of the class means to avoid being rejected (see figure d).

d) Four Thresholds for class pair (E vs C)



Vote for class 'E' if $t(4) < (D, X) < t(5)$

Vote for class 'O' if $t(1) < (D, X) < t(2)$

NO VOTE if $(D, X) \leq t(1)$ or $t(2) \leq (D, X) \leq t(4)$
or $(D, X) \geq t(5)$

=====

If 'E' is considered the first class and 'O' considered the second class of a class pair, CLPARS initially sets the discriminant threshold values to:

$$t(1) = \text{Mean}(O) - \text{delta}(E, O)/2$$

$$t(2) = \text{Mean}(O) + \text{delta}(E, O)/3$$

$$t(3) = \text{Mean}(O) + \text{delta}(E, O)/2$$

$$t(4) = \text{Mean}(E) - \text{delta}(E, O)/3$$

$$t(5) = \text{Mean}(E) + \text{delta}(E, O)/2$$

Where:

$t(1)$ = first threshold (5 thresholds total)

$\text{Mean}(i)$ = mean vector of class 'i'

$\text{delta}(i, j) = \text{Mean}(i) - \text{Mean}(j)$

*****)

FISHER (continued)

In this example the covariances of only those vectors that lie at the logic node are used, all the measurements are included in calculations, the minimum vote count is set to three, and the number of thresholds used is three.

Session with LONG prompts

DO YOU WANT TO USE ALL THE VECTORS OF
THE CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO' MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N)? /N/
DO YOU WISH TO ELIMINATE MEASUREMENTS
FROM THE COMPUTATION (Y/N)? /N/
ENTER THE MINIMUM VOTE COUNT,
(MAXIMUM IS NUMBER OF CLASSES MINUS 1) - /3/
ENTER THE NUMBER OF THRESHOLDS YOU WOULD LIKE (1-4) - /3/

(The program puts up the menu)

AD-A118 733

PAR TECHNOLOGY CORP NEW HARTFORD NY
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. USE--ETC(U)
JUN 82 S E HAEHN/ D MORRIS

F/G 9/2

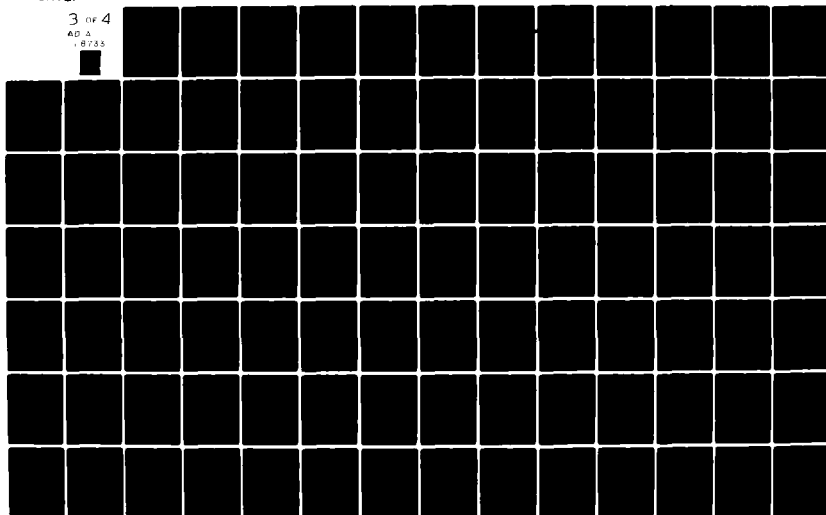
UNCLASSIFIED

PAR-82-21

NL

3 OF 4

AD A
8735



COMMAND NAME: FISHMOD

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

FISHMOD modifies pairwise logic at a user chosen logic node. The user has the option of modifying the minimum vote count and the number of thresholds.

This command can only be used after FISHER has created pairwise logic in the current logic tree.

To evaluate the logic use PWEVAL.

USER INTERACTION:

The user is asked (1) which pairwise logic node should be modified, (2) for a new minimum vote count, (3) for a new number of thresholds.

EXAMPLE(S):

In the following example, the user has decided to modify pairwise logic node number 3, change the minimum vote count to 6, and the number of thresholds to 2.

Session with SHORT prompts

LOGIC NODES WITH PAIRWISE LOGIC

```
2 3
NODE NUMBER? /3/
MINIMUM VOTE COUNT? /6/
NUMBER OF THRESHOLDS? /2/
```

(The program puts up the menu)

(PROMPT NOTES: *****)

There is an upper limit to the minimum vote count: one less than the number of classes present at the logic node.

*****)

FISHMCD (continued)

In this example the user decides to modify logic node 2, change the minimum vote count to 1, and the number of thresholds to 4.

Session with LONG prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3

ENTER THE LOGIC NODE YOU WISH TO MODIFY - /2/

ENTER THE MINIMUM VOTE COUNT,

(MAXIMUM IS NUMBER OF CLASSES MINUS 1) - /1/

ENTER THE NUMBER OF THRESHOLDS YOU WOULD LIKE (1-4) - /4/

(The program puts up the menu)

COMMAND NAME: HELP

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

HELP gives the user a helpful description of an OLPARS subject or of a specific OLPARS command. The user may also obtain a listing of all available help files. All help is written to the terminal.

USER INTERACTION:

The user types in a character string which may be one of the following:

- (1) an OLPARS command name
- (2) the name of an OLPARS subject
- (3) the initial characters (sub-string) of a command name or subject name
- (4) 'ALL'

Typing 'ALL' will give the user a listing of all available help files.

EXAMPLE(S):

Session with SHORT prompts

REQUEST? /DSCRMEAS/

DSCRMEAS computes the discriminant measurement evaluation statistics and outputs to the screen an overall ranking of the measurements for the current data set.

.
.
.

(DSCRMEAS help continued)

REQUEST? /PROMPTING/

OLPARS programs prompt the user in two modes: (1) short mode and (2) long mode. Long prompt mode gives the user more detailed instructions than short prompt mode. If the user is in short mode and needs more information, before (s)he can respond to a prompt, (s)he may type a question mark (?) and a long prompt will occur. To change the prompt mode, the user should run the command 'CDEFAULT.'

HELP (continued)

REQUEST? /<CR>/

(PROMPT NOTES: *****)

To leave the HELP command, the user must type the OLPARS 'exit command' character (<CR> in the examples given). If a legitimate filename cannot be created from the user-typed character string, or if the help file does not exist or cannot be opened, the user is notified that no help is available for the given request, and a prompt for another request appears.

*****)

Session with LONG prompts

In the following example, the user types a command for which there is no help available. (S)he then types 'ALL' to get a listing of available subjects. The listing in this example shows what an 'ALL' help file might look like.

TYPE IN SUBJECT STRING FOR HELP DESIRED
('ALL' FOR LIST OF AVAILABLE SUBJECTS) - /RANKORDER/
THERE IS NO HELP AVAILABLE FOR RANKORDER
TYPE IN SUBJECT STRING FOR HELP DESIRED
('ALL' FOR LIST OF AVAILABLE SUBJECTS) - /ALL/

| | | | |
|------------|------------|-----------|-----------|
| * ADDVEC | * ANYTHING | * APPEND | * BYEOLP |
| * CDEFAULT | CURRENTDS | CLASSLIST | * CCMNOD |
| * DRAWBNDY | EVALUATE | FEATURES | * FILEIN |
| * HELP | * L1EIGV | * L2EIGV | * LOGEVAL |
| MENU | * NAMELOG | * NMV | PLOTS |
| PROMPTING | * PRDTS | * RANK | * REDRAW |
| * S1EIGV | * S2EIGV | SCALING | * SCALZM |
| * SETDS | * SETLOG | * UNION | |

* INDICATES COMMAND NAMES

TYPE IN SUBJECT STRING FOR HELP DESIRED
('ALL' FOR LIST OF AVAILABLE SUBJECTS) - /<CR>/

COMMAND NAME: INTENSIFY

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

INTENSIFY highlights a class or classes currently displayed by drawing a solid outline around the given class distributions. It should be noted that (unless classes are well separated) when more than two classes are concurrently intensified, the display itself becomes cluttered with lines, which makes it difficult to observe the distributions. This routine is only applicable to one-space micro displays only. (If used on a one-space macro plot, a micro plot will be generated at the terminal.)

USER INTERACTION:

A class select list of valid display node names is displayed at the terminal. From these names you are asked to select valid class symbols for intensification. If a minus sign is entered as the first character in the input string, then all class symbols in the select list EXCEPT the ones entered will be intensified. If a star '*' is entered, then all the class symbols in the select list will be intensified. A minus-star '-*' will turn all of the class symbols off and there will be no classes intensified.

EXAMPLE(S):

In the following examples we will use the node names CORN, SOY, and RYE.

Session with SHORT prompts

CLASS SELECT LIST

CORN RYE SOY

CLASS SYMBOLS: /SC/

INTENSIFY (continued)

(PROMPT NOTES: *****)

There is no difference between the 'SHORT' and 'LONG' prompts. If a '?' is entered for the reply to prompt, you will receive an explanation at your terminal on how to reply to the prompt. If the class symbol entered is invalid, or is not in the select list, a message will be displayed at the terminal and you will receive the prompt again.

*****)

COMMAND NAME: L1ASLG

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L1ASLG projects a data set on to the Fisher direction associated with two algebraically-assigned groupings of data classes at an incomplete logic node. The groupings are determined by locating two classes whose mean vectors have the largest Euclidean separation. The remaining classes are associated with the class of this pair to which they are closest. If the class groupings are unacceptable, they may be regrouped manually. The final groupings need not comprise the entire data set; however, the entire data set is projected on the resulting Fisher vector.

USER INTERACTION:

The user is asked:

- 1) To select an incomplete logic node number (only when there is more than 1 incomplete logic node).
- 2) If all of the vectors of the classes which lie at the incomplete logic node are to be used in the computation of the Fisher and orthogonal discriminants (as opposed to using only those vectors which lie at the logic node).
- 3) If the computed division of the classes into two groups is not acceptable, the user is asked to select the classes to be found in groups 1 and 2.
- 4) If any measurements are to be eliminated from the computations.
- 5) If covariance or scatter matrices are to be used in the computations.

EXAMPLE(S):

The following examples assume that 2, 3, 4, and 5 are incomplete logic node numbers of the current logic. Node number 4 will be used. Only those vectors which lie at the incomplete logic node will be used in the computations. Classes ABCD, EFGH, IJKL, and MNOP lie at node number 4. Classes ABCD and MNOP are to be found in group

L1ASIG (continued)

1, and class IJKL is to be found in group 2. Scatter matrices will be used in the computations, and measurement number 3 will be eliminated.

Session with SHORT prompts

INCCMPLETE LOGIC NODES

2 3 4 5

LOGIC NODE? /4/

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC
NODE (Y/N)? /N/
INITIALIZING DISPLAY FILE HEADERS

<ERASE SCREEN>
CLASSES IN GROUP 1 -

RSTU

CLASSES IN GROUP 2 -

ABCD EF GH IJKL MNOP

ACCEPTABLE (Y/N)? /N/

<ERASE SCREEN>
SELECT THE CLASSES TO BE FOUND IN GROUP 1 -

CLASS SELECT LIST

ABCD EF GH IJKL MNOP RSTU

CLASS SYMBOLS: /AM/

SELECT THE CLASSES TO BE FOUND IN GROUP 2 -

CLASS SELECT LIST

ABCD EF GH IJKL MNOP RSTU

CLASS SYMBOLS: /I/

<ERASE SCREEN>
CLASSES IN GROUP 1 -

ABCD MNOP

CLASSES IN GROUP 2 -

IJKL

ACCEPTABLE (Y/N)? /Y/

ELIMINATE MEASUREMENTS (Y/N)? /Y/

MEAS = /3/

MEAS = /<CR>/

COVARIANCE/SCATTER MATRIX OPTION (C/S)? /S/

(PROMPT NOTES: *****)

If only one incomplete logic node lies at the current data set, then that is the logic node which will be used, and no selection will take place.

Class Symbols -

The class symbol string must be typed on a single line with no intermittent blanks, commas, and the like. If the class symbols typed are invalid class symbols, i.e. they don't exist, L1ASDG will prompt the user again for some valid class symbols. If the user types in one or more invalid class symbols, all valid class symbols must be retyped on the next try.

Group Division -

All classes at the logic node need not be used in the computations, but there must be at least one class in each group. Also, no class should appear in both groups 1 and 2. If a class is repeated, a message is printed and the user is reprompted for the classes in group 2.

*****)

Session with LONG prompts

INCOMPLETE LOGIC NODES

2 3 4 5

TYPE IN A LOGIC NODE NUMBER FROM THE INCOMPLETE LOGIC
NODE LIST - /4/

DO YOU WANT TO USE ALL THE VECTORS OF THE
CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO' MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N)? /N/

INITIALIZING DISPLAY FILE HEADERS

L1ASEG (continued)

<ERASE SCREEN>
CLASSES IN GROUP 1 -

RSTU

CLASSES IN GROUP 2 -

ABCD EFCH IJKL MNOP

IS THIS AN ACCEPTABLE DIVISION OF THE
CLASSES (Y/N)? /N/

<ERASE SCREEN>
SELECT THE CLASSES TO BE FOUND IN GROUP 1 -

CLASS SELECT LIST

ABCD EFCH IJKL MNOP RSTU

CLASS SYMBOLS: /AM/

SELECT THE CLASSES TO BE FOUND IN GROUP 2 -

CLASS SELECT LIST

ABCD EFCH IJKL MNOP RSTU

CLASS SYMBOLS: /I/

<ERASE SCREEN>
CLASSES IN GROUP 1 -

ABCD MNOP

CLASSES IN GROUP 2 -

IJKL

IS THIS AN ACCEPTABLE DIVISION OF THE
CLASSES (Y/N)? /Y/

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE
COMPUTATION?

(TYPE Y FOR YES, N FOR NO, <CR> TO EXIT) /Y/

MEASUREMENT NUMBER = /3/

MEASUREMENT NUMBER = /<CR>/

ARE COVARIANCE OR SCATTER MATRICES TO BE USED
IN THE COMPUTATION OF THE FISHER DISCRIMINANT
(C-COVARIANCE, S-SCATTER)? /S/

COMMAND NAME: L1CRDV

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L1CRDV allows a user to project the current data set onto a single coordinate for use in logic design. A one-space display is created at the terminal.

USER INTERACTION:

The user is asked:

- 1) To select an incomplete logic node number (only when there is more than 1 incomplete logic node).
- 2) Whether or not all of the vectors of the classes which lie at an incomplete logic are to be used in the logic calculations
- 3) The user is asked to select one coordinate (measurement number) for use in the projection.

EXAMPLE(S):

The following example shows the prompts obtained when only one incomplete logic node exists in a logic tree.

Session with SHORT prompts

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC
NODE (Y/N)? /Y/

COORDINATE NO. FOR THE X PROJECTION: /1/

INITIALIZING DISPLAY FILE HEADERS

(PROMPT NOTES: *****)

Coordinate Number -

The coordinate number entered by the user must be greater than zero and less than or equal to the vector dimensionality. If an invalid

L1CRLV (continued)

coordinate number is entered, an error message
will occur and the program will continue to
prompt the user for a valid coordinate number.

*****)

Session with LONG prompts

DO YOU WANT TO USE ALL THE VECTORS OF THE
CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO' MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N)? /N/

ENTER THE COORDINATE NUMBER TO USE FOR THE X
PROJECTION: /5/

INITIALIZING DISPLAY FILE HEADERS

COMMAND NAME: L1EIGV

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L1EIGV projects a data set onto an eigenvector for use in logic design. A brief description of the program algorithm follows:

- 1) If more than one incomplete logic node exists in the logic tree, you will be asked to specify one.
- 2) Eigenvalue computations will be based on only those classes that lie at the incomplete logic node. You may optionally choose, however, to base the eigenvalue computations on the covariance matrix of either all the vectors of the classes or only those vectors that lie at the incomplete logic node.
- 3) You will be given the option to eliminate any measurements from the projection.
- 4) The eigenvalues will be displayed in descending order at your terminal. You will select one eigenvalue number. The eigenvector which corresponds to the eigenvalue number you select will be used as a projection vector.
- 5) For each class at the incomplete logic node the chosen data vectors (see '2' above) will be projected on the projection vector, i.e.,

$X(i)$ = the projection of data vector i onto
projection vector 1

- 6) A macro plot or a micro plot will be displayed at your terminal.

USER INTERACTION:

You will be asked:

- 1) To select an incomplete logic node (if more than one exists).
- 2) If you want to use all vectors of the classes at the incomplete logic node or only those vectors that lie at the incomplete logic node.

L1EIGV (continued)

- 3) If any measurements are to be eliminated from from the computation of the projection vectors.
- 4) If you want a printout of the eigenvalues.
- 5) To select one eigenvalue for use in the projection.

EXAMPLE(S):

For both examples to follow, these conditions hold:

- 1) The vector dimensionality is 4.
- 2) The incomplete logic node numbers for the current data set are 1,2,3,4, and 5.

Session with SHORT prompts

INCCOMPLETE LOGIC NODES

1 2 3 4 5

LOGIC NODE? /1/

USE ALL VECTORS OF CLASSES AT INCCOMPLETE LOGIC NODE?
(Y/N)? /Y/

INITIALIZING DISPLAY FILE HEADERS

ELJMINATE MEASUREMENTS? Y/N /N/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 2.439089E+03 |
| 2 | 4.154599E+02 |
| 3 | 3.277070E+02 |
| 4 | 2.880631E+02 |

PRINTOUT? Y/N /N/

EIGENVECTOR NO. FOR THE X PROJECTION: /2/

(PROMPT NOTES: *****)

Measurement Number -

If the user types a carriage return the first time this prompt occurs, the program will exit. The following checks are made on the measurement number:

- 1) check for a valid measurement number (greater than zero and less than or equal to the vector dimensionality)
- 2) check to see that the user does not type the same measurement number twice
- 3) check to see that the user does not eliminate all the measurements

Note: errors from conditions 1 and 2 above cause the user to remain in prompt mode; an error from condition 3 causes an error message to be printed and the program to exit.

Eigenvector Number -

The program stays in prompt mode until the user types in a valid eigenvector number (greater than zero and less than or equal to the number of measurements used in the computation)

Printout? -

The program stays in prompt mode until the user types Y or N as the first character typed. If the user types Y a lineprinter copy of the above NUMBER/EIGENVALUE table is produced

*****)

L1EIGV (continued)

Session with LONG prompts

INCCOMPLETE LOGIC NODES

1 2 3 4 5

TYPE IN A LOGIC NODE NUMBER FROM THE INCCOMPLETE LOGIC
NODE LIST - /1/

DO YOU WANT TO USE ALL THE VECTORS OF
THE CLASSES THAT LIE AT THE INCCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO', MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N) /Y/

INITIALIZING DISPLAY FILE HEADERS

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE
COMPUTATION?
TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /Y/

MEASUREMENT NUMBER = /0/
INVALID MEASUREMENT NUMBER
VALID MEASUREMENT NUMBERS OCCUR BETWEEN 1 AND 4
MEASUREMENT NUMBER = /1/
MEASUREMENT NUMBER = /<CR>/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 1.878939E+03 |
| 2 | 3.958894E+02 |
| 3 | 2.931851E+02 |

DO YOU WANT A PRINTOUT OF THE EIGENVALUES?
TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /N/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE X
PROJECTION: /1/

COMMAND NAME: L2ASDG

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L2ASDG projects a data set on to the optimal discriminant plane associated with two algebraically-assigned groupings of data classes at an incomplete logic node. The groupings are determined by locating two classes whose mean vectors have the largest Euclidean separation. The remaining classes are associated with the class of this pair to which they are closest. If the class groupings are unacceptable, they may be regrouped manually. The final groupings need not comprise the entire data set; however, the entire data set is projected on the resulting optimal discriminant plane.

USER INTERACTION:

The user is asked:

- 1) To select an incomplete logic node number (only when there is more than 1 incomplete logic node).
- 2) If all of the vectors of the classes which lie at the incomplete logic node are to be used in the computation of the fisher and orthogonal discriminants (as opposed to using only those vectors which lie at the logic node).
- 3) If the computed division of the classes into two groups is not acceptable, the user is asked to select the classes to be found in groups 1 and 2.
- 4) If any measurements are to be eliminated from the computations.
- 5) If covariance or scatter matrices are to be used in the computations.

EXAMPLE (S):

The following examples assume that 2, 3, 4, and 5 are incomplete logic node numbers at the current data set. Node number 4 will be used. Only those vectors which lie at the incomplete logic node will be used in the computations. Classes ABCD, EFGH, IJKL, and MNOP lie at node number 4. Classes ABCD and MNOP are to be found in group

L2ASIG (continued)

1, and class IJKL is to be found in group 2. Scatter matrices will be used in the computations, and measurement number 3 will be eliminated.

Session with SHORT prompts

INCCMPLETE LOGIC NODES

2 3 4 5

LOGIC NODE? /4/

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC
NODE (Y/N)? /N/

INITIALIZING DISPLAY FILE HEADERS

<ERASE SCREEN>

CLASSES IN GROUP 1 -

RSTU

CLASSES IN GROUP 2 -

ABCD EF GH IJKL MNOP

ACCEPTABLE (Y/N)? /N/

<ERASE SCREEN>

SELECT THE CLASSES TO BE FOUND IN GROUP 1 -

CLASS SELECT LIST

ABCD EF GH IJKL MNOP RSTU

CLASS SYMBOLS: /AM/

SELECT THE CLASSES TO BE FOUND IN GROUP 2 -

CLASS SELECT LIST

ABCD EF GH IJKL MNOP RSTU

CLASS SYMBOLS: /I/

<ERASE SCREEN>

CLASSES IN GROUP 1 -

ABCD MNOP

CLASSES IN GROUP 2 -

IJKL

ACCEPTABLE (Y/N)? /Y/

ELIMINATE MEASUREMENTS (Y/N)? /Y/

MEAS = /3/

MEAS = /<CR>/

COVARIANCE/SCATTER MATRIX OPTION (C/S)? /S/

(PROMPT NOTES: *****)

If only one incomplete logic node lies at the current data set, then that is the logic node which will be used, and no selection will take place.

Class Symbols -

The class symbol string must be typed on a single line with no intermittent blanks, commas, and the like. If the class symbols typed are invalid class symbols, i.e. they don't exist, L2ASDG will prompt the user again for some valid class symbols. If the user types in one or more invalid class symbols, all valid class symbols must be retyped on the next try.

Group Division -

All classes at the logic node need not be used in the computations, but there must be at least one class in each group. Also, no class should appear in both groups 1 and 2. If a class is repeated, a message is printed and the user is reprompted for the classes in group 2.

*****)

Session with LONG prompts

INCOMPLETE LOGIC NODES

2 3 4 5

TYPE IN A LOGIC NODE NUMBER FROM THE INCOMPLETE LOGIC NODE LIST - /4/

DO YOU WANT TO USE ALL THE VECTORS OF THE CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE IN LOGIC CALCULATIONS

('NO' MEANS USE ONLY THOSE VECTORS THAT LIE AT THE LOGIC NODE) (Y/N)? /N/

INITIALIZING DISPLAY FILE HEADERS

L2ASDG (continued)

<ERASE SCREEN>
CLASSES IN GROUP 1 -

RSTU

CLASSES IN GROUP 2 -

ABCD EFGH IJKL MNOP

IS THIS AN ACCEPTABLE DIVISION OF THE
CLASSES (Y/N)? /N/

<ERASE SCREEN>
SELECT THE CLASSES TO BE FOUND IN GROUP 1 -

CLASS SELECT LIST

ABCD EFGH IJKL MNOP RSTU

CLASS SYMBOLS: /AM/

SELECT THE CLASSES TO BE FOUND IN GROUP 2 -

CLASS SELECT LIST

ABCD EFGH IJKL MNOP RSTU

CLASS SYMBOLS: /I/

<ERASE SCREEN>
CLASSES IN GROUP 1 -

ABCD MNOP

CLASSES IN GROUP 2 -

IJKL

IS THIS AN ACCEPTABLE DIVISION OF THE
CLASSES (Y/N)? /Y/

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE
COMPUTATION?

(TYPE Y FOR YES, N FOR NO, <CR> TO EXIT) /Y/

MEASUREMENT NUMBER = /3/

MEASUREMENT NUMBER = /<CR>/

ARE COVARIANCE OR SCATTER MATRICES TO BE USED
IN THE COMPUTATION OF THE FISHER DISCRIMINANT
(C-COVARIANCE, S-SCATTER)? /S/

COMMAND NAME: L2CRDV

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L2CRDV allows a user to project the current data set onto two coordinates for use in logic design. A two-space display is created at the terminal.

USER INTERACTION:

The user is asked:

- 1) To select an incomplete logic node number (only when there is more than one incomplete logic node).
- 2) Whether or not all of the vectors of the classes which lie at an incomplete logic node are to be used in the logic calculations.
- 3) To select two coordinates (measurement numbers) for use in the projection.

EXAMPLE(S):

The following example shows the prompts obtained when only one incomplete logic node exists in a logic tree:

Session with SHORT prompts

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC
NODE (Y/N)? /Y/

COORDINATE NO. FOR THE X PROJECTION: /6/

COORDINATE NO. FOR THE Y PROJECTION: /1/

INITIALIZING DISPLAY FILE HEADERS

(PROMPT NOTES: *****)

Coordinate Number -

The coordinate number entered by the user must be greater than zero and less than or equal to the vector dimensionality. If an invalid coordinate number is entered, an error message

will occur and the program will continue to
prompt the user for a valid coordinate number.

*****)

The following example shows the prompts obtained when
the incomplete logic node numbers at the current data
set are 1,2,3,4, and 5.

Session with LONG prompts

INCOMPLETE LOGIC NODES

1 2 3 4 5

TYPE IN A LOGIC NODE NUMBER FROM THE INCOMPLETE LOGIC
NODE LIST - /5/

DO YOU WANT TO USE ALL THE VECTORS OF
THE CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO' MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N)? /N/

ENTER THE COORDINATE NUMBER TO USE FOR THE X
PROJECTION: /6/

ENTER THE COORDINATE NUMBER TO USE FOR THE Y
PROJECTION: /1/

INITIALIZING DISPLAY FILE HEADERS

COMMAND NAME: L2EIGV

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L2EIGV projects a data set onto a pair of eigenvectors for use in logic design. A brief description of the program algorithm follows:

- 1) If more than one incomplete logic node exists in the logic tree, you will be asked to specify one.
- 2) Eigenvalue computations will be based on only those classes that lie at the incomplete logic node. You may optionally choose, however, to base the eigenvalue computations on the covariance matrix of either all the vectors of the classes or only those vectors that lie at the incomplete logic node.
- 3) You will be given the option to eliminate any measurements from the projection.
- 4) The eigenvalues will be displayed in descending order at your terminal. You will select two eigenvalue numbers. The two eigenvectors which correspond to the two eigenvalue numbers you select will be used as projection vectors.
- 5) For each class at the incomplete logic node the chosen data vectors (see '2' above) will be projected on the projection vectors, i.e.,
$$X(i) = \text{the projection of data vector } i \text{ onto projection vector 1}$$
$$Y(i) = \text{the projection of data vector } i \text{ onto projection vector 2}$$
- 6) A scatter plot or a cluster plot will be displayed at your terminal.

L2EIGV (continued)

USER INTERACTION:

You will be asked:

- 1) To select an incomplete logic node (if more than one exists).
- 2) If you want to use all vectors of the classes at the incomplete logic node or only those vectors that lie at the incomplete logic node.
- 3) If any measurements are to be eliminated from the computation of the projection vectors.
- 4) If you want a printout of the eigenvalues.
- 5) To select two eigenvalues for use in the projection.

EXAMPLE(S):

For both examples to follow, these conditions hold:

- 1) The vector dimensionality is 4.
- 2) The incomplete logic node numbers for the current data set are 1,2,3,4, and 5.

Session with SHORT prompts

INCOMPLETE LOGIC NODES

1 2 3 4 5

LOGIC NODE? /1/

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC NODE?

(Y/N)? /Y/

INITIALIZING DISPLAY FILE HEADERS

ELIMINATE MEASUREMENTS (Y/N)? /Y/

MEAS = /3/

MEAS = /<CR>/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 1.725496E+03 |
| 2 | 4.118130E+02 |
| 3 | 2.986911E+02 |

PRINTOUT? Y/N /Y/

EIGENVECTOR NO. FOR THE X PROJECTION: /2/

EIGENVECTOR NO. FOR THE Y PROJECTION: /3/

(PROMPT NOTES: *****)

Measurement Number -

If the user types a carriage return the first time this prompt occurs, the program will exit. The following checks are made on the measurement number:

- 1) check for a valid measurement number (greater than zero and less than or equal to the vector dimensionality)
- 2) check to see that the user does not type the same measurement number twice
- 3) check to see that the user does not eliminate all the measurements

Note: errors from conditions 1 and 2 above cause the user to remain in prompt mode; an error from condition 3 causes an error message to be printed and the program to exit.

Eigenvector Number -

The program stays in prompt mode until the user types in a valid eigenvector number (greater than zero and less than or equal to the number of measurements used in the computation)

Printout? -

The program stays in prompt mode until the user types Y or N as the first character typed. If the user types Y a lineprinter copy of the above NUMBER/EIGENVALUE table is produced

*****)

L2EIGV (continued)

Session with LONG prompts

INCOMPLETE LOGIC NODES

1 2 3 4 5

TYPE IN A LOGIC NODE NUMBER FROM THE INCOMPLETE LOGIC
NODE LIST - /1/

DO YOU WANT TO USE ALL THE VECTORS OF
THE CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO', MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N) /Y/

INITIALIZING DISPLAY FILE HEADERS

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE
COMPUTATION?

TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /N/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 2.439089E+03 |
| 2 | 4.154599E+02 |
| 3 | 3.277070E+02 |
| 4 | 2.880631E+02 |

DO YOU WANT A PRINTOUT OF THE EIGENVALUES?
TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /N/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE X
PROJECTION: /1/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE Y
PROJECTION: /2/

COMMAND NAME: L2FSHP

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

L2FSHP projects the vectors at an incomplete logic node on two fisher directions which correspond to two pairs of data classes within the selected data set. Note, the resultant fisher discriminant vectors are ortho-normalized before they are used for data projection.

USER INTERACTION:

The user is asked:

- 1) To select an incomplete logic node number (only when there is more than 1 incomplete logic node).
- 2) If all of the vectors of the classes which lie at the incomplete logic node are to be used in the computation of the fisher discriminants (as opposed to using only those vectors which lie at the logic node)
- 3) To select a pair of classes to be used in the computation of the first fisher discriminant.
- 4) To select a pair of classes to be used in the computation of the second fisher discriminant.
- 5) If covariance or scatter matrices are to be used in the computations.
- 6) If any measurements are to be eliminated from the computations.

EXAMPLE(S):

The following examples assumes that 1,2,3,4, and 5 are incomplete logic node numbers at the current data set. Node number 4 will be used. Only those vectors which lie at the incomplete logic node will be used in the computations. Classes ABCD,EFGH,IJKL, and MNOP lie at node number 4. Classes ABCD and MNOP will be used to compute fisher discriminant 1 and EFGH and MNOP will be used to compute fisher discriminant 2. Scatter matrices will be used in the computations, and measurement number 3 will be eliminated.

Session with SHCRT prompts

INCOMPLETE LOGIC NODES

1 2 3 4 5

LOGIC NODE? /4/

USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC

NODE (Y/N)? /N/

INITIALIZING DISPLAY FILE HEADERS

SELECT TWO CLASSES FROM THE LIST BELOW TO USE IN
THE COMPUTATION OF FISHER 1

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /AM/

SELECT TWO CLASSES FROM THE LIST BELOW TO USE IN
THE COMPUTATION OF FISHER 2

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /EM/

COVARIANCE/SCATTER MATRIX OPTION (0/1)? /1/

ELIMINATE MEASUREMENTS (Y/N)? /Y/

MEAS = /3/

MEAS = /<CR>/

(PROMPT NOTES: *****)

If only one incomplete logic node lies at the current
data set, then that is the logic node which will be used,
and no selection will take place.

If the two fisher discriminants are linearly dependent,
the message 'THE TWO FISHER PROJECTION VECTORS ARE
LINEARLY DEPENDENT. THE X AND Y PROJECTION VECTORS WILL
BE THE SAME VECTOR.' is printed.

Class Symbols -

The class symbol string must be typed on a single
line with no intermittent blanks, commas, and the
like. If the class symbols typed are invalid class
symbols, i.e. they don't exist, L2FSHP will prompt
the user again for some valid class symbols. If the
user types in one or more invalid class symbols, all
valid class symbols must be retyped on the next try.

*****)

Session with LCNG prompts

INCCOMPLETE LOGIC NODES

1 2 3 4 5

TYPE IN A LOGIC NODE NUMBER FROM THE INCCOMPLETE LOGIC
NODE LIST - /4/

DO YOU WANT TO USE ALL THE VECTORS OF THE
CLASSES THAT LIE AT THE INCCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NC' MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N)? /N/

INITIALIZING DISPLAY FILE HEADERS

FROM THE LIST OF LOWEST NODES BELOW, SELECT THE
FIRST AND SECOND CLASS TO BE USED TO COMPUTE
FISHER 1 (IE. CS)

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /AM/
FROM THE LIST OF LOWEST NODES BELOW, SELECT THE
FIRST AND SECOND CLASS TO BE USED TO COMPUTE
FISHER 2 (IE. CS)

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /EM/

ARE COVARIANCE OR SCATTER MATRICES TO BE USED
IN THE COMPUTATION OF THE FISHER DISCRIMINANT
(0-COVARIANCE, 1-SCATTER)? /1/

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE
COMPUTATION?
(TYPE Y FOR YES, N FOR NO, <CR> TO EXIT) /Y/
MEASUREMENT NUMBER = /3/
MEASUREMENT NUMBER = /<CR>/

LISTLOGS

COMMAND NAME: LISTLOGS

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

LISTLOGS displays at the users terminal, the names of all the user's logic trees, whether or not the logic is complete, and the design sets of those logics with their dimensionality.

USER INTERACTION: NONE

EXAMPLE(S):

This is an example of the display with logic trees present. The display is in the form:

```
<logic name> <treename>.<nodename (class)>
```

The number in parenthesis () is the dimensionality of the design set. The (I) represents an incomplete logic.

| LOGIC TREES | DESIGN DATA SET | DATE: dd-mmm-yy hh:mm:ss |
|-------------|-------------------|-----------------------------|
| LOGNAME (I) | TREENAME.**** (4) | |
| LOGIC2 | NASA1 .Clov (4) | |
| LOGIC3 (I) | TREE3 .soy (4) | |
| HWOOD | HOLLY .WOOD (11) | |

(NOTES: *****)

In the above example, '****' represents the senior (highest) node in the data tree.

A message is printed at the user's terminal if there are not any logic trees in the file.

*****)

LISTTREES

COMMAND NAME: LISTTREES

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

LISTTREES displays at the users terminal, the names of all the user's data trees along with their dimensionality.

USER INTERACTION: NONE

EXAMPLES(S):

This is an example of the display with trees present. The number in parenthesis () is the dimensionality of the tree.

USER DATA TREES DATE: dd-mmm-yy hh:mm:ss

NAME2 (4)
NAME3 (4)
TESTNAME (4)

(NOTES: *****)

A message is printed at the user's terminal if there are not any data trees in the file.

*****)

LOGEVAL

COMMAND NAME: LOGEVAL

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

LOGEVAL enables the user to test any logic, complete or incomplete, against a data set or a single vector having the same dimensionality. The logic used is the current logic and the data set used is the current data set. LOGEVAL operates in four modes: (1) complete, (2) restore, (3) classifier, and (4) single vector.

In complete mode, a complete logic is evaluated and a confusion matrix display is created. Restore mode assures that the logic design commands know which vectors currently reside at which logic nodes. Restoring an incomplete logic allows the user to continue logic design on that logic after a different logic has been designed or evaluated using the same data set. In classifier mode, a complete or incomplete logic is evaluated and a table is created listing the number of vectors residing at each lowest logic node. In single vector mode, a single vector is passed through the logic, and the result is displayed on the screen.

USER INTERACTION:

The user is asked to select the mode of operation for LOGEVAL - either complete, restore, classifier, or single vector. If the user has run the command REASNAME (see note on following page), and if complete, classifier (with a complete logic), or single vector mode is chosen, a query for use of the reassociated class names during logic evaluation will be given. If the operator selects complete mode, a query will be given to determine if the user wants a printout of the confusion matrix and/or an error (misclassification) listing. If classifier mode is selected, the user is asked if a printout of the tabular output and/or a printout of vector id's and the logic nodes to which the vectors were assigned is desired. When single vector mode is selected, the operator is asked (1) if the vector to use is from the current data set or will be typed in at the terminal, and (2) to type in either the class symbol and vector id of the vector or the vector itself, depending on the response to (1).

.....

NOTE

If the current data set is not the data set that was used to design the current logic, or if the current data set does not have the same classes as the data set that was used to design the logic, the user should execute the command REASNAME. REASNAME will allow the user to rename the classes in the logic to be the same as the classes in the current data set. The renamed classes are called reassocated classes. The user should use the reassocated class names whenever the data and logic tree class names differ.

EXAMPLE(S):

Session with SHORT prompts

LOGEVAL OPERATION MODES:

- (1) COMPLETE
- (2) RESTORE
- (3) CLASSIFIER
- (4) SINGLE VECTOR

(NOTE: An example will be given for each operation mode. The examples assume that reassocated class names exist for the current logic.)

.....

OPERATION MODE: /1/

USE REASSOCIATED CLASS NAMES (Y/N)? /Y/

(a confusion matrix is displayed.)

CONFUSION MATRIX PRINTOUT (Y/N)? /Y/

(a confusion matrix is printed at the lineprinter.)

ERROR LISTING (Y/N)? /Y/

(an error listing is printed at the lineprinter.)

.....

OPERATION MODE: /2/

(no more prompts occur.)

LOGEVAL (continued)

OPERATION MODE: /3/

USE REASSOCIATED CLASS NAMES (Y/N)? /Y/

(a classification table containing the lowest logic node numbers, associated class (or INCOMPLETE or **** (which represents 'reject')), and vector count is displayed.)

PRINTOUT (Y/N)? /Y/

(the table described above is printed at the lineprinter.)

LISTING OF VECTOR ID'S AND LOGIC NODES (Y/N)? /Y/

(a table containing the vector id's, the logic node numbers, and the associated class (or INCOMPLETE or **** (reject)) is printed at the lineprinter.)

.....

OPERATION MODE: /4/

USE REASSOCIATED CLASS NAMES (Y/N)? /Y/

(1) USE A VECTOR FROM THE CURRENT DATA SET

(2) TYPE IN THE VECTOR AT THE TERMINAL

SELECT AN OPTION (1 OR 2): /1/

CLASS SYMBOL OF THE VECTOR TO BE EVALUATED: /A/

VECTOR ID: /24/

THE VECTOR WAS ASSIGNED TO LOGIC NODE 5 CLASS ANOD

(PROMPT NOTES: *****)

In single vector mode, measurements may be typed in either integer, floating point, or exponential format.

*****)

Session with LONG prompts

LOGEVAL OPERATES IN THE FOLLOWING FOUR MODES:

(1) COMPLETE: THE CURRENT DATA SET IS EVALUATED USING A COMPLETED LOGIC, AND A CONFUSION MATRIX IS DISPLAYED.

(2) RESTORE: AN INCOMPLETE LOGIC, WITH A DESIGN DATA SET NAME THAT IS THE SAME AS THE CURRENT DATA SET NAME, IS USED TO EVALUATE THE CURRENT DATA SET AND RESTORE THE VECTORS TO THE CORRECT LOGIC NODES.

(3) CLASSIFIER: THE CURRENT DATA SET IS EVALUATED, USING EITHER A COMPLETE OR AN INCOMPLETE LOGIC, TO DETERMINE THE NUMBER OF VECTORS ASSIGNED TO EACH CLASS IN THE LOGIC.

(4) SINGLE VECTOR: A SINGLE VECTOR (WHICH MAY COME FROM THE CURRENT DATA SET OR MAY BE TYPED IN BY THE USER) IS EVALUATED USING A COMPLETED LOGIC AND THE RESULT IS DISPLAYED AT THE TERMINAL.

(NOTE: An example will be given for each operation mode. The examples assume there are no reassociated class names for the current logic.)

.....
SELECT THE MODE OF OPERATION (1 THRU 4)
TO USE FOR LOGIC EVALUATION: /1/

(a confusion matrix is displayed.)

DO YOU WANT A CONFUSION MATRIX PRINTOUT (Y/N)? /N/

DO YOU WANT A LISTING OF MISCLASSIFIED VECTORS (Y/N)? /N/

.....
SELECT THE MODE OF OPERATION (1 THRU 4)
TO USE FOR LOGIC EVALUATION: /2/

(no more prompts occur.)

LOGEVAL (continued)

SELECT THE MODE OF OPERATION (1 THRU 4)
TC USE FOR LOGIC EVALUATION: /3/

(a table containing the lowest logic node numbers,
associated class (or INCCMPLETE or **** (reject)),
and vector count is displayed.)

DO YOU WANT A LISTING OF THE ABOVE TABLE (Y/N)? /N/

DO YOU WANT A LISTING OF VECTOR ID'S AND THE LOGIC NODES
TO WHICH VECTORS WERE ASSIGNED (Y/N)? /N/

.....

(NOTE: In the following example, the user has chosen to
type in the vector at the terminal. The vector
must have the same dimensionality as the design
data set of the logic. Prompting for
measurements will continue until the user has
typed in the correct number of measurements.)

SELECT THE MODE OF OPERATION (1 THRU 4)
TC USE FOR LOGIC EVALUATION: /4/

OPTIONS:

(1) USE A VECTOR FROM THE CURRENT DATA SET

(2) TYPE IN THE VECTOR AT THE TERMINAL

TYPE A '1' OR A '2' DEPENDING ON THE DESIRED OPTION: /2/

THE VECTOR MUST HAVE 2 MEASUREMENTS

MEASUREMENTS SHOULD BE TYPED ONE PER LINE

TYPE IN MEASUREMENT 1 OF THE VECTOR: /10/

TYPE IN MEASUREMENT 2 OF THE VECTOR: /20/

THE VECTOR WAS ASSIGNED TO LOGIC NODE 5 CLASS ANOD

LTRENAME

COMMAND NAME: LTRENAME

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

Change the name of an CLPARS logic tree.

USER INTERACTION:

The name of the logic tree to be changed is requested from the user, along with the new name of the tree.

EXAMPLE(S):

The following example shows the tree 'GRAINS' being renamed to 'CEREALS'.

Session with SHORT prompts

FROM? /GRAINS/
TO? /CEREALS/

(PROMPT NOTES: *****)

If the new name of the tree is already in use, the user is requested for permission to destroy the existing tree.

*****)

In the following example, the tree 'CEREALS' already exists.

Session with LONG prompts

ENTER NAME OF TREE TO BE RENAMED - /GRAIN1/
ENTER NEW NAME OF TREE - /CEREALS/
THE NAME 'CEREALS' IS IN USE IN YOUR DIRECTORY;
DO YOU WANT TO DESTROY THE LOGIC WITH THAT NAME (Y/N)?
/Y/

MAKETREE

COMMAND NAME: MAKETREE

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

MAKETREE is used to create a new data tree from nodes of existing data trees.

To COMBINE trees:

MAKETREE can create a new data tree with copies of the lowest nodes of up to ten trees. In the process, if any display symbols are duplicated, new display symbols will be substituted (display symbols of lowest data nodes must always be unique).

To MERGE trees:

MAKETREE can create a new data tree by merging all of the vectors from similarly-named nodes of various trees into nodes (with the same names) in the new tree. A maximum of ten trees can be merged.

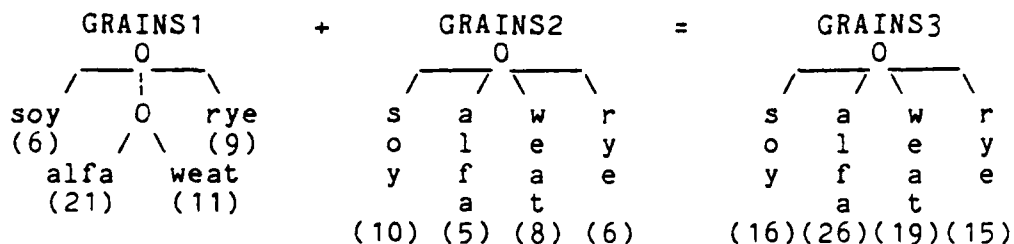
USER INTERACTION:

The user is asked:

1. for the name of the new data tree
2. whether to combine or merge existing data trees
3. for the number of trees to use in the creation
4. for the names of existing data trees to be used
- (5. When merging trees:
 - a. if vector identifiers are to be resequenced
 - b. if printer listing is desired of changed vector identifiers)

EXAMPLE(S):

In the following example, the data trees GRAINS1 and GRAINS2 are going to be merged into the new tree, GRAINS3 (see diagram). Note that the node names and the number of lowest nodes (4) are identical in both GRAINS1 and GRAINS2 (GRAINS1 has a total of 6 nodes, including the intermediate nodes (marked 'O'), while GRAINS2 has only 5; this example shows that the structure of the two data trees being merged is irrelevant to the merging operation). In the diagram, the numbers in parenthesis represent the number of vectors residing at each of the data nodes.



Session with SHORT prompts

 TREENAME FOR NEW TREE? /GRAINS3/

THE DATA TREE CAN BE CREATED BY
 COMBINING (C) OR MERGING (M)
 NODES FROM EXISTING TREES.

YOUR CHOICE? /M/
 NUMBER OF TREES? /2/
 TREENAME? /GRAINS1/
 TREENAME? /GRAINS2/
 RESEQUENCE VECTOR IDENTIFIERS (Y/N)? /Y/
 PRINT NEW SEQUENCE (Y/N)? /Y/

(PROMPT NOTES: *****)

After the 'new tree name' prompt has been answered and the given tree is found to exist, MAKETREE asks the user if the tree is to be destroyed.

If a component data tree does not exist, MAKETREE indicates this and reprompts the user for another name.

When '1' is specified as the number of component trees to make up the new data tree, the user is effectively making a copy of the original data tree. In this case, the 'merge mode' is the suggested method of tree

MAKETREE (continued)

creation because it will be execute faster than the 'combine mode'.

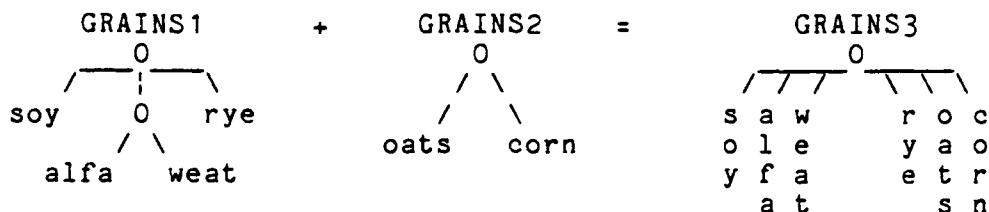
NOTE

The resequence output obtained from merging vectors shows which trees contributed vectors to the new data tree, from which class the vectors came, the old vector identifier, and the new vector identifier.

(e.g., Vectors from data tree 'GRAINS1'
CLASS (soy) PREVIOUS ID.= 23 NEW ID.= 5)

*****)

In the following example, the two data trees, GRAINS1 and GRAINS2, are to be combined to form the new data tree, GRAINS3 (see diagram). The symbol '0', occurring by itself, represents an intermediate data node (i.e., not a lowest node). The data structure of the component trees is irrelevant to MAKETREE.



Session with LONG prompts

ENTER A NAME FOR THE NEW TREE THAT YOU ARE CREATING.
(8 CHARS. MAX.) - /GRAINS3/

MODES OF OPERATION THAT CAN BE USED

- (COMBINE) ALL LOWEST NODES OF DIFFERENT TREES
ARE PLACED UNDER THE SENIOR NODE OF
THE NEW TREE
- (MERGE) SAME-NAMED LOWEST NODES OF DIFFERENT TREES
ARE MERGED TOGETHER AND PLACED UNDER THE
SENIOR NODE OF THE NEW TREE

YOUR CHOICE? /C/

ENTER THE NUMBER OF TREES TO BE USED

IN MAKING UP THE NEW TREE - /2/

ENTER THE NAME OF A TREE TO BE USED IN MAKING
UP THE NEW TREE - /GRAINS1/

ENTER THE NAME OF A TREE TO BE USED IN MAKING
UP THE NEW TREE - /GRAINS2/

COMMAND NAME: MATRIX

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

MATRIX deals with the deletion and entering of transformation matrices into the saved transformation matrix (SM) file. MATRIX displays a list of options and asks the user to select one. By selecting the appropriate option, the user may (1) delete a previously saved matrix, (2) print a line printer listing of one or all previously saved matrices, (3) dump at the terminal one or all previously saved matrices, (4) list at the terminal the matrix descriptions of all previously saved matrices, or (5) enter a new matrix into the file.

USER INTERACTION:

The user is asked to select an option number from the list of options. The additional interaction for each option is as follows:

- Option 1: The user is asked for the name of the matrix to be deleted. An asterisk (*) will indicate that all saved matrices are to be deleted.
- Option 2: The user is asked for the name of the matrix to be printed at the line printer. An asterisk (*) will indicate that all saved matrices are to be printed.
- Option 3: The user is asked for the name of the matrix to be dumped at the terminal. An asterisk (*) will indicate that all saved matrices are to be dumped.
- Option 4: No additional interaction.
- Option 5: The user is asked for the new matrix's name, type (either normal or eigenvector transformation matrix), dimensionality, and from where the matrix is to be entered (either directly from the terminal or from a file). If the matrix is to be entered from a file, the user is asked for the file name. Otherwise, the matrix is to be entered one row vector at a time.

MATRIX (continued)

If a program has terminated abnormally, or the program is such that the information on the screen should not be erased immediately, the user will be asked if (s)he wants to continue. If so, the option list will be redisplayed, if not, the program will end. Otherwise, the option list will automatically be redisplayed and the user will be prompted for an option number.

EXAMPLE (S):

The examples which appear below will follow a logical order most likely to be used when entering a matrix and maintaining the SM file.

The first example will involve entering (directly from the terminal) the sample eigenvector transformation matrix which follows:

$$\text{MATRIX1} = \begin{bmatrix} 2.0 & 1.0 & 0.0 & 3.2 & 4.3 \\ 1.0 & 2.0 & 3.0 & 4.0 & 5.0 \\ 1.2 & 4.9 & 7.0 & 8.5 & 6.1 \end{bmatrix}$$

Example (2) will list the descriptions of all matrices in the file, making sure that the matrix was successfully saved, and assuming that the SM file was empty before MATRIX1 was entered. Example (3) will display the entries of MATRIX1 at the screen. Example (4) will involve making a line printer listing of MATRIX1. Example (5) will delete MATRIX1.

After the first example below, the option list will be indicated by the line (OPTION LIST).

Session with SHORT prompts

=====

EXAMPLE 1:

<ERASE SCREEN>
SAVED TRANSFORMATION MATRICES

1. DELETE
2. PRINT
3. DUMP
4. LIST
5. ENTER

OPTION NUMBER? /5/

<ERASE SCREEN>
ENTER A MATRIX

MATRIX NAME? /MATRIX1/
TYPE OF TRANSFORMATION? (N/E) /E/
DIMENSION? /5/
MATRIX ENTERED FROM TERMINAL OR FILE? (T/F) /T/

<ERASE SCREEN>
ENTER THE MATRIX

ROW 1> 2.0 1.0 0.0 3.2 4.3

ROW 2> 1.0 2.0 3.0 4.0 5.0

ROW 3> 1.2 4.9 7.0 8.5 6.1

ROW 4> <CR>

<ERASE SCREEN>
(OPTION LIST)

(PRCMT NOTES: *****)

If the SM file is full (i.e., contains ten saved matrices), the message 'THERE IS NO AVAILABLE SPACE LEFT IN THE SM FILE TO ENTER A NEW MATRIX' will be printed and the user will be asked if (s)he wants to continue.

MATRIX NAME

The matrix name should be from 1 to 8 characters in length, where the first character is alphabetic and the rest are alphanumeric. The name must also be unique within the file.

MATRIX (continued)

MATRIX ENTERED FROM A FILE

If the matrix is to be entered from a file, the file must already exist and contain only the matrix to be saved.

MATRIX ENTERED FROM THE TERMINAL

If the matrix is to be entered directly from the terminal, and the user asks to quit before any elements have been entered, the message 'NO MATRIX HAS BEEN SAVED' is printed and the user is asked if (s)he wants to continue.

If the user asks to quit before the present row vector has been filled to the specified dimension, the vector is filled with zeros and the next row prompt appears.

If the user asks to quit directly after a row prompt appears, this is a normal termination. All previous vectors have been saved, and the option list is displayed.

*****)

=====

EXAMPLE 2:

(OPTION LIST)

OPTION NUMBER? /4/

<ERASE SCREEN>

| MATRIX NAME | NO. OF ROWS | DIMENSION |
|-------------|-------------|-----------|
|-------------|-------------|-----------|

| | | |
|---------|---|---|
| MATRIX1 | 3 | 5 |
|---------|---|---|

CONTINUE? /Y/

<ERASE SCREEN>

(OPTION LIST)

(PROMPT NOTES: *****)

If the SM file is empty, the message 'THERE ARE NO PREVIOUSLY SAVED MATRICES IN THE FILE' will be typed and the user will be asked if (s)he wants to go on.

*****)

=====

=====

EXAMPLE 3:

(OPTION LIST)
OPTION NUMBER? /3/

<ERASE SCREEN>
DUMP SAVED MATRIX

MATRIX NAME? /MATRIX1/

<ERASE SCREEN>
NAME = 'MATRIX1' NO. OF ROWS = 3 DIMENSION = 5

MATRIX (LISTED BY ROW VECTORS) -

ROW 1> 2.0 1.0 0.0 3.2 4.3

ROW 2> 1.0 2.0 3.0 4.0 5.0

ROW 3> 1.2 4.9 7.0 8.5 6.1

CONTINUE? /Y/

<ERASE SCREEN>
(OPTION LIST)

(PROMPT NOTES: *****)

If the SM file is empty, the message 'THERE ARE NO PREVIOUSLY SAVED MATRICES IN THE FILE' will be printed and the user will be asked if (s)he wants to continue.

If a non-existing matrix name has been entered, the message printed is 'MATRIX ---- HAS NOT BEEN FOUND IN THE FILE' where ---- is the matrix name input by the user. The user will be reprompted for the name.

If the screen is full and the whole matrix has not yet been displayed, the user will be asked if (s)he wants to see the next page. After a complete matrix has been displayed, if all matrices are to be shown, the user is asked if (s)he wants to see the next matrix. If only one matrix was to be seen, the user is asked if (s)he wants to continue.

*****)

MATRIX (continued)

=====

EXAMPLE 4:

(OPTION LIST)
OPTION NUMBER? /2/

<ERASE SCREEN>
PRINT SAVED MATRIX

MATRIX NAME? /MATRIX1/

<ERASE SCREEN>
(OPTION LIST)

(PROMPT NOTES: *****)

If the SM file is empty, the message 'THERE ARE NO PREVIOUSLY SAVED MATRICES IN THE FILE' will be typed and the user will be asked if (s)he wants to go on.

If a non-existing matrix name has been entered, the message printed is 'MATRIX ---- HAS NOT BEEN FOUND IN THE FILE'. The user will be reprompted for the name.

*****)

=====

EXAMPLE 5:

(OPTION LIST)
OPTION NUMBER? /1/

<ERASE SCREEN>
DELETE SAVED MATRIX

MATRIX NAME? /MATRIX1/

<ERASE SCREEN>
(OPTION LIST)

(PROMPT NOTES: *****)

If the SM file is empty, the message 'THERE ARE NO SAVED MATRICES IN THE FILE' will be printed and the user will be asked if (s)he wants to continue.

If a non-existing matrix name has been entered, the message printed is 'MATRIX ---- HAS NOT BEEN FOUND IN THE FILE'. The user will be reprompted for the name.

*****)

=====

=====

Session with LONG prompts

Due to the length of this program description,
no examples will be given with long prompts.
However, the option list with long prompts is
as follows:

SAVED TRANSFORMATION MATRICES

1. DELETE - DELETE A SAVED MATRIX FROM THE FILE.
2. PRINT - MAKE A LINE PRINTER LISTING OF THE NAME,
TYPE, DIMENSION, AND ENTRIES OF A SAVED
MATRIX.
3. DUMP - DISPLAY AT THE TERMINAL THE NAME, TYPE,
DIMENSION, AND ENTRIES OF A SAVED MATRIX.
4. LIST - DISPLAY AT THE TERMINAL THE NAMES, TYPES,
AND DIMENSIONS OF ALL SAVED MATRICES IN
THE FILE.
5. ENTER - SAVE A MATRIX.

OPTION NUMBER? /<CR>/

=====

MATXFRM

COMMAND NAME: MATXFRM

CATEGORY: TRANSFORMATION COMMAND

FUNCTIONAL DESCRIPTION:

MATXFRM will perform either a normal, eigenvector or matrix transformation using a saved matrix chosen by the user from the saved transformation matrix (SM) file to create a new data tree. The type of saved matrix will determine which type of transformation will be performed.

The transformation itself will involve matrix multiplication between the current data set and the specified saved (or transformation) matrix. By the definition of matrix multiplication, the dimension of the current data set and the transformation matrix must be equal.

If an eigenvector or matrix transformation is to be performed, the dimension of the newly created data tree will be equal to the number of vectors found in the transformation matrix. Note, if there is only 1 vector in the transformation matrix, the resultant tree will have scalar values for vector, mean, and covariance entities.

USER INTERACTION:

The user is asked for the name of a saved transformation matrix and a new tree name.

EXAMPLE(S):

The following examples assume that MATRIX1 is the name of the saved transformation matrix chosen for use in the computations and NEWTREE is the name of the new tree to be created.

Session with SHORT prompts

MATRIX NAME? /MATRIX1/

NEW TREE NAME? /NEWTREE/

(PROMPT NOTES: *****)

Matrix Name

The matrix name entered must correspond to an existing matrix stored in the saved transformation matrix (SM) file. If no such matrix exists, the user will be reprompted for another matrix name.

If an eigenvector transformation matrix has been chosen, and the number of eigenvalues used to compute this matrix is 1, the message 'BECAUSE OF THE DIMENSIONALITY OF THE CHOSEN SAVED MATRIX, THE RESULT OF THE EIGENVECTOR TRANSFORMATION WILL BE A SIMPLE SCALAR VALUE' is printed and the user is asked if (s)he wants to continue.

When a new data tree has successfully been created, the message printed is 'DATA TREE '----' HAS SUCCESSFULLY BEEN CREATED' where ---- is the name of the new data tree.

*****)

Session with LONG prompts

ENTER THE NAME OF THE SAVED MATRIX TO
BE USED IN THE MATRIX TRANSFORMATION - /MATRIX1/

TYPE IN NAME TO GIVE TO THE OLPARS DATA
TREE THAT WILL BE CREATED AS A RESULT
OF THE TRANSFORMATION (8 CHARS. MAX.) - /NEWTREE/

MEASXFRM

COMMAND NAME: MEASXFRM

CATEGORY: TRANSFORMATION COMMAND (system dependent)

FUNCTIONAL DESCRIPTION:

MEASXFRM gives the user the capability of transforming the current data set into a new data set. This transformation takes the form of FORTRAN statements which are a function of the measurements from the original data set (e.g., $NM(1) = OM(1) + OM(2)$ states that measurement 1 of the new data set equals the sum of measurements 1 and 2 of the old data set). The user creates a FORTRAN subroutine called XFORM using the following template:

```
      SUBROUTINE XFORM (OM, OLNATH, NM, NLNGTH)
C=====
C THIS IS A MEASUREMENT TRANSFORMATION SUBROUTINE
C USED BY THE OLPARS 'MEASXFRM' COMMAND.
C
C (NOTE, WHEREVER THE SYMBOL OF TWO QUESTION MARKS
C  APPEAR, THE DIMENSION OF THE NEW VECTOR MUST BE
C  SUPPLIED BY THE USER.)
C=====
      INTEGER OLNATH, NLNGTH
      REAL    OM(OLNATH), NM(??)
C
C      (( TELL MEASXFRM THE SIZE OF THE NEW DATA SET )))
      IF(NLNGTH .GT. 0) GOTO 1000
      NLNGTH = ??
      RETURN
C      ENDIF
1000  CONTINUE
C
C      ** USER SUPPLIED VECTOR TRANSFORMATION
C      SHOULD APPEAR BELOW **
C
C      ,
C      ,
C      transformation statements
C      ,
C
C      ** USER SUPPLIED VECTOR TRANSFORMATION
C      SHOULD APPEAR ABOVE **
C
C      RETURN
      END
```

=====

where

OM represents the old measurement vector of length
OLNGTH, and

NM represents the new measurement vector of length
NLNGTH.

NLNGTH is initially an output variable (tells MEASXFRM
the size of the new vectors). On all subsequent calls
to XFORM; OM, OLNGTH, and NLNGTH are considered input
variables, while NM is an output variable of XFORM.

Note, MEASXFRM can be used in excess measurement mode.

USER INTERACTION:

User is asked:

1. For the name of the transformation.
(name of the file containing an 'XFORM' subroutine)
2. If the transformation exists.
 - a. If the transformation is to be altered.
 - b. For the name of the editor to be invoked
to edit the existing transformation.
3. (When a new transformation is being created)
 - a. For the number of measurements per vector
in the new data set.
 - b. For the name of the editor to be invoked
to edit the new transformation.
 - c. If the compilation was succesful.
 - d. If the task build (linking) was succesful.
4. Name of the data tree being created.
5. If the transformation is to be saved.

MEASXFRM (continued)

EXAMPLE(S):

In the following example, the current data set, GRAINS, contains 12 measurements; the first of which is really a scaling factor for the rest of the measurements. The resulting tree, XGRAINS, will only contain 11 measurements after the transformation takes place.

```
>*ENTER TRANSFORMATION NAME [S R:0-9]: /FORGRAINS/  
>*DOES TRANSFORMATION ALREADY EXIST (Y/N)? [S R:0-1]: /N/  
>*ENTER DIMENSIONALITY OF NEW DATA SET [D R:0-150]: /11/  
>*TYPE IN NAME OF EDITOR TO BE USED [S]: /EDI/  
> EDI FORGRAINS.FTN
```

(user edits file to place in desired transformation)

```
> F4P FORGRAINS,FORGRAINS/-SP=FORGRAINS/-WR/-TR/-CK  
>*WAS COMPILATION SUCCESFUL (Y/N)? [S R:0-1]: /Y/  
;  
; The following task build takes approximately  
; 2 minutes on a PDP 11/45  
;  
> TKB @FORGRAINS.BLD  
>*TASK BUILD SUCCESSFUL (Y/N)? [S R:0-1]: /Y/  
> PIP FORGRAINS.BLD;*,FORGRAINS.ODL;*/DE  
> RUN FORGRAINS
```

NEW TREE NAME? /XGRAINS/

TRANSFORMATION COMPLETE

```
>*SHOULD TRANSFORMATION BE SAVED (Y/N)? [S R:0-1]: /N/
```

(PROMPT NOTES: *****)

In the above example, the lines beginning with '>*' are prompts given by the DEC command processor found on RSX11M operating systems. A command file is being executed to guide the operation of 'MEASXFRM'.

If at any time a prompt receives the OLPARS standard exit response, the command file will exit, as expected. The command file, however, does not have the usual long and short prompt convention (the resulting task does have the long-short prompt convention).

The '[S R:number-number]' at the end of each prompt tells the user that an alphanumeric string (S) (or a decimal number (D), as in the dimensionality prompt) is expected by the prompt, with a length (or numeric value as in 0-150) in the specified range (R).

The editor can be any editor that is available on the local operating system.

The RSX11M command file expects the F4P FORTRAN compiler supplied by Digital Equipment Corp. The slashes (/) that occur in that line are not surrounding user responses, but switches that the compiler uses to modify its standard mode of operation.

*****)

MOVEC

COMMAND NAME: MOVEC

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

MOVEC permits the user to reorient data vectors by moving their projected points on a two-space coordinate vector scatter display, which is currently displayed at the terminal.

This routine is intended for use with trees which contain nearest neighbor reference patterns.

USER INTERACTION:

User identifies the vector to be moved by placing the graphics cursor 'over' the corresponding display character and typing in any alphabetic character (except 'Q' or 'R').

After the vector is identified, the user moves the graphics cursor to the position the vector is to be placed, and types in another alphabetic character (except 'Q').

The character 'Q', typed after the first prompt, terminates (quits) the process. The character 'R' redisplayes the current display.

The character 'Q', typed after the second prompt, sends the program back to the first prompt to start vector selection over (i.e., the current vector chosen is not the one to be moved). The character 'R', typed after the second prompt, places the vector at its new location and redisplayes the current display.

EXAMPLE(S):

In the example which follows please note the meaning of two symbols:

... means 'position the graphics cursor'
K means 'any alphanumeric key'
(except 'Q' or 'R')

CHOOSE ONE /...K/
NEW LOCALE /...K/
CHOOSE ONE /Q/

(PROMPT NOTES: *****)

In the example, the 'CHOOSE ONE' prompt is given to notify the operator that it is time to choose the vector to be moved. The operator moves the graphics cursor to the position of the vector that is going to be reoriented (i.e., 'over' the vector display symbol found on the terminal display) and types in an alphanumeric character. A rectangle is drawn around the display symbol of the vector that is closest to the given point. In the example, the correct vector was chosen by MOVEC. Therefore, when the 'NEW LOCALE' prompt appeared, the operator placed the cursor to the desired vector position and entered another alphanumeric character. (The vector display symbol of the moved vector is immediately displayed to the user at the newly requested position.)

If no vector can be found at the user's requested position, (s)he will be notified by a 'NOT FOUND' message in the lower left hand portion of the display.

*****)

NAMELOG

COMMAND NAME: NAMELOG

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

NAMELOG creates a new logic tree and makes it the 'current logic' being used.

USER INTERACTION:

The user is asked the name of the new logic to be created. If the tree already exists, the user is asked if the tree is to be destroyed.

EXAMPLE(S):

In the following example the new logic tree 'NEWLOGIC' is created and becomes the 'current logic'.

Session with SHORT prompts

TREENAME? /NEWLOGIC/

(PROMPT NOTES: *****)

The treename that is entered by the user must be a 'legal' treename. NAMELOG will print out an error message if this condition is not met and prompting will continue until a valid treename is entered.

*****)

In the following example the tree 'NEWLOGIC' already exists and the user decides to destroy it.

Session with LONG prompts

ENTER THE NEW LOGIC TREE NAME (MAXIMUM 8 CHARACTERS)
/NEWLOGIC/

THE NAME "NEWLOGIC"
IS IN YOUR DIRECTORY;
DO YOU WISH TO DESTROY THE LOGIC WITH THAT NAME (Y/N)?
/Y/

COMMAND NAME: NMEVAL

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

NMEVAL evaluates nearest mean vector logic at a user chosen logic node. (See NMV for additional information).

This command can only be used after NMV has created logic in the current logic tree, or NMVMOD has modified existing NMV logic.

NOTE, If this command is used twice at the same logic node, without deleting the existing logic, a confusion matrix of zeroes will be created.

USER INTERACTION:

The user is asked which NMV logic node should be evaluated.

EXAMPLE(S):

In the following example, the user has decided to evaluate vectors at NMV logic node number 3.

Session with SHORT prompts

LOGIC NODES WITH NMV LOGIC

2 3
NODE NUMBER? /3/

(Confusion matrix is displayed.)

(The program puts up the menu)

In this example the user decides to evaluate vectors at logic node 3.

Session with LONG prompts

LOGIC NODES WITH NMV LOGIC

2 3
ENTER THE LOGIC NODE YOU WISH TO EVALUATE - /3/

(Confusion matrix is displayed.)

(The program puts up the menu)

NMV

COMMAND NAME: NMV

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

NMV creates nearest mean vector logic for the current data set. That is, when logic evaluation occurs, each data vector will be assigned to the class whose mean is closest to that vector.

One of four methods for computing a "distance" from the vector to each class mean can be chosen. They are the Euclidean distance, a class variance weighted distance, Mahalanobis distance, and a Quadratic Classifier*.

A rejection region may be assigned to each class (mean). When a vector is being classified, the distance between it and the nearest mean of a class must be less than the reject distance of that class. Otherwise, the vector is not associated with the class (it is rejected). The reject region is chosen in units of standard deviations. Note, reject distances are not available for the Quadratic Classifier because this metric is not in terms of distance.

To evaluate the newly created logic, use NMEVAL.

* See the paper entitled 'Classification of Atypical Cells in the Automatic Screening for Cervical Cancer' by Oliver et. al. in the IEEE Proceedings of the Conference on Pattern Recognition and Image Processing, May 31 - June 2, 1978, pp. 476-482.

USER INTERACTION:

The user is asked (1) if all vectors from the classes which lie at the logic node should be used, (2) whether there are measurements to be ignored, (3) which distance option is desired, and (4) which reject strategy to use.

EXAMPLE(S):

In the following example, the user has decided to use only those vectors which lie at the logic node, to ignore the second measurement, to use the weighted euclidean distance, with no reject regions.

Session with SHORT prompts

```
-----
USE ALL VECTORS OF CLASSES AT INCOMPLETE LOGIC NODE? /N/
IGNORE MEASUREMENTS (Y/N)? /Y/
MEASUREMENT # (END WITH 0)? /2/
MEASUREMENT # (END WITH 0)? /0/
DISTANCE OPTION (1-4)? /2/
```

REJECT DISTANCE OPTIONS

- 1) NO REJECT DISTANCES
- 2) OVERALL MULTIPLIER (WITH EACH CLASS STD)
- 3) SEPARATE MULTIPLIERS (FOR EACH CLASS STD)
- 4) OVERALL MULTIPLIER (WITH THE AVERAGE CLASS STD)

REJECT DISTANCE OPTION - /1/

(The program puts up the menu)

(PROMPT NOTES: *****)

The user must make sure that the last ignore measurement prompt entered is a zero, and not <CR> or the program will exit.

Reject Distance Notes

"Overall" multiplier with each class standard deviation means that the value you type in is applied to each class standard deviation. If you type in a value of two (2) then vectors lying outside the region represented by two standard deviations from the nearest class mean will be rejected.

"Separate" multipliers for each class standard deviation means that you can select a standard deviation constant to be applied to each class. For instance, if there are three classes, A, B, and C, you can select a reject region of 1 standard deviation to be applied to class A, of 2.5 to be associated with class B, and one of 3.5 to be applied to class C.

"Overall" multiplier with the average class standard deviation means that only a single value is applied to the average value of all the class standard deviations, that is, if there are two classes, A and B, and the standard deviation of class A is 25 and the standard deviation of class B is 15, then the multiplier typed in will be applied to the average of these two values, 20. This calculated standard deviation value is used for

NMV (continued)

all classes. With this option the reject distance is the same distance for all classes.

If separate multipliers are desired for the reject regions, they are requested by the program one at a time. Beware of requesting help while entering the multipliers, for this wipes out all multipliers entered and begins the entire reject sequence again.

*****)

In this example the user decides to use only those vectors that lie at the logic node, ignore no measurements, use the Mahalanobis distance option, with a reject distance of 2.5 standard deviations.

Session with LONG prompts

DO YOU WANT TO USE ALL THE VECTORS OF
THE CLASSES THAT LIE AT THE INCOMPLETE LOGIC NODE
IN LOGIC CALCULATIONS
('NO' MEANS USE ONLY THOSE VECTORS THAT
LIE AT THE LOGIC NODE) (Y/N)? /N/
DO YOU WISH TO HAVE SOME MEASUREMENTS IGNORED (Y/N) - /N/
ENTER THE OPTION YOU WOULD LIKE

- 1 - EUCLIDEAN DISTANCE
 - 2 - WEIGHTED VECTOR
 - 3 - MAHALANOBIS DISTANCE
 - 4 - QUADRATIC CLASSIFIER
- /3/

REJECT DISTANCES MAY BE ASSIGNED TO THE CLASSES. WHEN A VECTOR IS BEING CLASSIFIED, THE DISTANCE BETWEEN IT AND THE NEAREST MEAN OF A CLASS MUST BE LESS THAN THE REJECT DISTANCE OF THAT CLASS. OTHERWISE THE VECTOR IS REJECTED. THE REJECT MULTIPLIER(S) ENTERED ARE APPLIED TO THE STANDARD DEVIATION (STD) OF EACH CLASS AS INDICATED BELOW.

REJECT DISTANCE OPTIONS

- 1) NO REJECT DISTANCES
- 2) OVERALL MULTIPLIER (WITH EACH CLASS STD)
- 3) SEPARATE MULTIPLIERS (FOR EACH CLASS STD)
- 4) OVERALL MULTIPLIER (WITH THE AVERAGE CLASS STD)

REJECT DISTANCE OPTION - /2/
ENTER THE MULTIPLIER - /2.5/

(The program puts up the menu)

COMMAND NAME: NMVMOD

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

NMVMOD modifies nearest mean vector logic at a user chosen logic node. The user has the choice of modifying the distance option and/or the reject criteria.

This command can only be used after NMV has created logic in the current logic tree.

To evaluate the logic use NMEVAL.

USER INTERACTION:

The user is asked (1) which NMV logic node should be modified, (2) which distance option is desired, and (3) which reject strategy to use.

EXAMPLE(S):

In the following example, the user has decided to modify NMV logic node number 3, and use the weighted euclidean distance, with no reject regions.

Session with SHORT prompts

LOGIC NODES WITH NMV LOGIC

2 3
NODE NUMBER? /3/
DISTANCE OPTION (1-4)? /2/
REJECT DISTANCE OPTIONS

- 1) NO REJECT DISTANCES
- 2) OVERALL MULTIPLIER (WITH EACH CLASS STD)
- 3) SEPARATE MULTIPLIERS (FOR EACH CLASS STD)
- 4) OVERALL MULTIPLIER (WITH THE AVERAGE CLASS STD)

REJECT DISTANCE OPTION - /1/

(The program puts up the menu)

NMVMOD (continued)

(PROMPT NOTES: *****)

If separate multipliers are desired for the reject regions, they are requested by the program one at a time. Beware of requesting help while entering the multipliers, for this wipes out all multipliers entered and begins the reject sequence over.

*****)

In this example the user decides to modify logic node 3, use the mahalanobis distance option, with an overall reject multiplier of 2.5 standard deviations.

Session with LONG prompts

LOGIC NODES WITH NMV LOGIC

2 3

ENTER THE LOGIC NODE YOU WISH TO MODIFY - /3/

ENTER THE OPTION YOU WOULD LIKE

- 1 - EUCLIDEAN DISTANCE
- 2 - WEIGHTED VECTOR
- 3 - MAHALANOBIS DISTANCE
- 4 - QUADRATIC CLASSIFIER

- /3/

REJECT DISTANCES MAY BE ASSIGNED TO THE CLASSES. WHEN A VECTOR IS BEING CLASSIFIED, THE DISTANCE BETWEEN IT AND THE NEAREST MEAN OF A CLASS MUST BE LESS THAN THE REJECT DISTANCE OF THAT CLASS. OTHERWISE THE VECTOR IS REJECTED. THE REJECT MULTIPLIER(S) ENTERED ARE APPLIED TO THE STANDARD DEVIATION (STD) OF EACH CLASS AS INDICATED BELOW

REJECT DISTANCE OPTIONS

- 1) NO REJECT DISTANCES
- 2) OVERALL MULTIPLIER (WITH EACH CLASS STD)
- 3) SEPARATE MULTIPLIERS (FOR EACH CLASS STD)
- 4) OVERALL MULTIPLIER (WITH THE AVERAGE CLASS STD)

REJECT DISTANCE OPTION - /2/

ENTER THE MULTIPLIER

- /2.5/

(The program puts up the menu)

COMMAND NAME: NNMOD

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

NNMOD modifies nearest neighbor logic at a user chosen logic node. The user has the option of changing the

- 1) number of neighbors being used during evaluation,
- 2) the reference pattern tree to be used during evaluation, and
- 3) the set of measurements to be ignored during evaluation.

This command can only be used after the NRSTNBR command has created nearest neighbor logic in the current logic tree.

USER INTERACTION:

The user is asked:

1. which nearest neighbor (NN) logic node should be changed (providing there is more than one NN logic in the current logic tree).
2. for the modification option to be used
 - (a) neighbor count modification -
(queried for neighbor count)
 - (b) use of different reference pattern tree -
(queried for tree name)
 - (c) measurements to be ignored -
(asked if any measurements are to be ignored and for index of measurement to be ignored)

NNMOD (continued)

EXAMPLE(S):

In the following example, the operator has finished creating nearest neighbor logic by using the NRSTNBR logic design command. However, the number of nearest neighbors desired during a test set evaluation is three (NRSTNBR only uses 1 nearest neighbor). Also, measurement 4 of the test set is to be ignored during evaluation.

Session with SHORT prompts

-
1. CHANGE NEIGHBOR COUNT (CURRENT = 1)
 2. CHANGE REFERENCE PATTERN TREE NAME (CURRENT = cnntest)
 3. CHANGE MEASUREMENTS IGNORED (ALL MEAS. IN USE)

MODIFICATION OPTION (1-3)? /1/
NEIGHBOR COUNT (1-9)? /3/

(screen erased)

1. CHANGE NEIGHBOR COUNT (CURRENT = 3)
2. CHANGE REFERENCE PATTERN TREE NAME (CURRENT = cnntest)
3. CHANGE MEASUREMENTS IGNORED (ALL MEAS. IN USE)

MODIFICATION OPTION (1-3)? /3/
IGNORE MEASUREMENTS (Y/N)? /Y/
MEASUREMENT NO. (END WITH 0)? /4/
MEASUREMENT NO. (END WITH 0)? /0/

(screen erased)

1. CHANGE NEIGHBOR COUNT (CURRENT = 3)
2. CHANGE REFERENCE PATTERN TREE NAME (CURRENT = cnntest)
3. CHANGE MEASUREMENTS IGNORED

MODIFICATION OPTION (1-3)? /<CR>/

(PROMPT NOTES: *****)

The current number of neighbors to use in evaluation, and the current reference pattern tree name can be seen in parenthesis to the right of the corresponding option. The message '(ALL MEAS. IN USE)' will be found to the right of the change-measurements-ignored option only when there are no measurements currently being ignored.

If a 'no' answer is given to the ignore-measurements prompt, all measurements are used in subsequent evaluations.

When a new reference pattern tree is going to be used, NNMOD checks to see if the tree exists and if the number of measurements in the reference pattern tree is identical to that of the design data set. NNMOD does not verify that the classes in the reference pattern tree are identical to the classes present at the logic node. This check is made at logic evaluation time.

The OLPARS standard command exit response (<CR>) is the only way NNMOD can be halted.

*****)

The next example shows the operator requesting a different reference pattern to be used during a subsequent logic evaluation.

Session with LONG prompts

1. CHANGE NEIGHBOR COUNT (CURRENT = 3)
2. CHANGE REFERENCE PATTERN TREE NAME (CURRENT = cnntest)
3. CHANGE MEASUREMENTS IGNORED

MODIFICATION OPTION (1-3)? /2/
 TYPE IN NAME OF NEAREST NEIGHBOR REFERENCE PATTERN TREE
 TO BE USED IN SUBSEQUENT EVALUATIONS - /REFPAT/

(screen erased)

1. CHANGE NEIGHBOR COUNT (CURRENT = 3)
2. CHANGE REFERENCE PATTERN TREE NAME (CURRENT = REFPAT)
3. CHANGE MEASUREMENTS IGNORED

MODIFICATION OPTION (1-3)? /<CR>/

NORMXFRM

COMMAND NAME: NORMXFRM

CATEGORY: Transformation Command

FUNCTIONAL DESCRIPTION:

NORMXFRM creates a new data tree which is a normalized version of the current data set. As a result of the transformation, the variance of each measurement in the new data tree becomes one.

USER INTERACTION:

The user is prompted for the name of the OLPARS data tree to be created. After the transformation is complete, the user is asked if s(he) wishes to save the transformation matrix (vector). If the user responds with 'Y' (for yes), s(he) is then asked to enter a name for the transformation matrix to be saved.

EXAMPLE(S):

Session with SHORT prompts

The following example assumes the user has a tree in her/his directory called 'NEWTREE'. The user decides to destroy the existing tree 'NEWTREE' so that the name 'NEWTREE' can be used for the tree resulting from the transformation. The example also assumes that the matrix 'MATRIXA' exists in the SM file.

```
NEW TREE NAME? /NEWTREE/  
THE NAME "NEWTREE" IS IN USE; DESTROY (Y/N)? /Y/  
TRANSFORMATION COMPLETE  
SAVE TRANSFORMATION MATRIX (Y/N)? /Y/  
MATRIX NAME? /MATRIXA/  
MATRIX 'MATRIXA' ALREADY EXISTS IN THE FILE  
MATRIX NAME? /MATRIXB/
```

(PROMPT NOTES: *****)

Tree Name -

If the tree name entered by the user is the same as the current tree name, an error message is printed and 'NORMXFRM' remains in prompt mode. If the tree name entered already exists, but is not the same as the current tree name, the user is asked if s(he) wants to destroy the existing tree. If the user types 'Y' (for yes), 'NORMXFRM' creates a new tree with the user-specified name. If the user types 'N' (for no), 'NORMXFRM' asks for another tree name.

Saved Matrix Name -

If the user wants to save the transformation matrix, and there is no available space in the Saved Matrix (SM) File, a message is printed and NORMXFRM exits. (NOTE: Although the normalization procedure was successful, the user will be unable to save the vector used in this transformation. Try again later after deleting some saved matrices from the SM File.) The matrix name entered cannot exceed 8 characters of which the first must be alphabetic and the remainder alphanumeric. The matrix name must also be unique within the SM file. The user is prompted until a valid and unique matrix name is entered.

*****)

Session with LONG prompts

The following example assumes the name of the current data tree is 'NASA'

TYPE IN NAME TO GIVE TO THE OLPARS DATA TREE
 THAT WILL BE CREATED AS A RESULT OF THE TRANSFORMATION
 (8 CHARS. MAX.) - /NASA/
 NEW TREE NAME MUST BE DIFFERENT
 FROM CURRENT DATA SET TREE NAME
 TYPE IN NAME TO GIVE TO THE OLPARS DATA TREE
 THAT WILL BE CREATED AS A RESULT OF THE TRANSFORMATION
 (8 CHARS. MAX.) - /NEWNASA/
 TRANSFORMATION COMPLETE
 DO YOU WISH TO SAVE THE TRANSFORMATION MATRIX (Y/N)? /Y/
 TYPE IN THE NAME OF THE MATRIX TO BE SAVED - /MATRIX/

ERROR IN MATRIX NAME. THE FIRST CHARACTER MUST
 BE ALPHABETIC, AND THE REST ALPHANUMERIC.

TYPE IN THE NAME OF THE MATRIX TO BE SAVED - /MATRIX1/

NRSTNBR

COMMAND NAME: NRSTNBR

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

NRSTNBR creates nearest neighbor logic at a specified node of the current logic tree. The resultant logic consists of a reference pattern tree created by using

- 1) the entire current data tree,
- 2) mean vectors (from the current data tree) of the classes present at the logic node, or
- 3) the "Condensed Nearest Neighbor" rule of Hart.*

No partial evaluation is performed for this logic. Partial evaluations are not needed for the CNN rule, or when using the entire data tree, because classification is defined to be 100 percent for the design data classes present at the logic node.

(When using the mean vector rule, logic evaluation should give results similar to Nearest Mean Vector (NMV) logic with the Euclidean distance measure.)

For the CNN rule, statistics are presented to the user (e.g., number of passes, number of vectors from classes present at the logic node, and number of reference patterns selected).

* See HART, P. E., "The Condensed Nearest Neighbor Rule,"
IEEE Trans. Info. Th., vol. IT 14, pp. 515-516, May 1968

USER INTERACTION:

- o Select logic node (if more than one incomplete logic node exists in the logic tree).
- o Select reference pattern tree generation rule.
- o Name reference pattern tree (if rule dictates creation of new tree)
- o If CNN rule is chosen:
 - oo Select interaction at each iteration of rule.
 - oo Abort generation of NN logic
- o Select measurements to be ignored during evaluation

EXAMPLE(S):

In the following example, nearest neighbor logic is being created at the senior node of the logic tree. Since there is only one node in the logic tree, the user is not asked to select a logic node. A decision was made to use the entire current data tree as the nearest neighbor reference pattern tree. Consequently, there is no prompt for the user to specify a reference pattern tree name. All measurements are to be used during any subsequent logic evaluation.

Session with SHORT prompts

REFERENCE PATTERN SELECTION

1. ENTIRE DATA TREE
2. MEAN VECTORS
3. CNN RULE

(1-3)? /1/
IGNORE MEASUREMENTS (Y/N)? /N/

(PROMPT NOTES: *****)

If the reference pattern selection rule was either 2 or 3, a new 'data' tree would have been created to contain the reference pattern vectors (a tree name prompt would have appeared; If the name given already existed, the user is queried for permission to destroy the tree).

Note, when not using the 'entire data tree' rule, the resultant reference pattern tree will only contain vectors from the classes present at the logic node. If the entire data tree is used, all the classes in the current data tree will be in the reference pattern tree. Thus, there may be more reference pattern classes than classes present at the logic node. The only problem this may cause during subsequent logic evaluation is slower execution time.

If the condensed nearest neighbor rule is chosen, the ability to review each 'pass' of reference pattern generation is allowed. After each pass, a prompt appears, asking if logic creation should be completed. Note, if no new reference pattern vectors are chosen from the design data tree, no 'review' prompt is given because the logic is finished.

*****)

In the next example the condensed nearest neighbor rule was picked to generate the reference pattern tree, 'NNREFPAT'. Since 'NNREFPAT' was in existence, permission to over write the older version of the tree was granted. Because this was the first time nearest neighbor logic was being used on the particular data set, it was not clear that the CNN rule would create a 'good' reference pattern tree. Therefore, interaction with each CNN rule 'pass' was requested. The first measurement was not useful in separating the data classes, so it was ignored.

Session with LONG prompts

REFERENCE PATTERNS ARE TO BE GENERATED FROM ...

1. THE ENTIRE DESIGN DATA TREE
2. THE MEAN VECTORS OF THE DESIGN DATA SET
3. THE CONDENSED NEAREST NEIGHBOR RULE

(1-3)? /3/

TYPE IN NAME TO GIVE TO REFERENCE PATTERN TREE -
/NNREFPAT/

THE NAME 'NNREFPAT' IS IN USE IN YOUR DIRECTORY;
DO YOU WANT TO DESTROY THE DATA SET WITH THAT NAME
(Y/N)? /Y/

DO YOU WANT THE ABILITY TO ABORT THE CONDENSED
NEAREST NEIGHBOR RULE AFTER EACH ITERATION (Y/N)? /Y/

DO YOU WISH TO HAVE SOME MEASUREMENTS IGNORED (Y/N)? /Y/
ENTER THE MEASUREMENT NUMBER (END WITH 0) - /1/
ENTER THE MEASUREMENT NUMBER (END WITH 0) - /0/

(erase screen)

12 VECTORS IN CLASSES PRESENT. - ANSWER (Y/N) TO 'OK?'

PASS 1 (3 NEW REFERENCE PATTERNS, TOTAL OF 3) - OK? /Y/
PASS 2 (0 NEW REFERENCE PATTERNS, TOTAL OF 3)

COMMAND NAME: OPTIMLMOD

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

OPTIMLMOD is used to modify logic at a user chosen pairwise logic node. In this case OPTIMLMOD creates (or changes existing) optimal discriminate plane logic at the node. The Fisher direction is combined with a computed optimal orthogonal direction to create an optimal discriminating plane. The vectors are projected onto this plane and the user is asked to enter a boundary, which will be used to discriminate between classes in an evaluation. The vote is for one class or the other, i. e., there is no excess region. The only way a vector can be rejected is if it fails to acquire the minimum vote count.

This command can only be used after FISHER has created pairwise logic in the current logic tree.

To evaluate the logic, use PWEVAL.

USER INTERACTION:

The user is asked (1) which pairwise logic node should be modified, (2) which class pair should be modified, (3) for a boundary, (4) for the symbol of the class associated with the convex region.

EXAMPLE(S):

(In the examples which follow please note the meaning of two symbols:

... means 'position the graphics cursor'
K means 'any alphanumeric key
except 'Q''

The current entry mode (BOUNDARY, CONVEX PT., CONVEX SYMBOL) is shown to the left of the user input.)

In the following example, the user has decided to modify pairwise logic node number 3, in particular, the class pair of 'soy' and 'oats'. When satisfied, a comma is entered.

OPTIMLMOD (continued)

Session with SHORT prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3

NODE NUMBER? /3/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS

-OR- '*' FOR NEXT CLASS PAIR

-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /cs/

(The projected vectors are displayed.)

BOUNDARY: /...K...K...K...KQ/

CONVX PT: /...K/

CONVEX SYMBOL? /c/

To create a boundary consisting of only one line
segment:

BOUNDARY: /...K...KQ/

CONVX PT: /...K/

CONVEX SYMBOL? /c/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS

-OR- '*' FOR NEXT CLASS PAIR

-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /,/

(The program puts up the menu)

(PROMPT NOTES)*****

When exiting from the class pair select prompt, a ',' is different from a <CR>. The ',' will save the changes and cause the option to be changed to OPTIMLMOD. A <CR> is interpreted as the standard OLPARS exit. Note: A 'comma exit' will always change the current option, even if no modification has been made.

The convex symbol requested is the display symbol of the class located in the user declared convex region.

The next class pair symbol will cause OPTIMLMOD to set the pair index to the following pair in this series...

(1,2), (1,3), (1,4), ... (1,TruCls), (2,3), (2,4), etc.

If the pointer is at the end of the list, the next pair is the first pair (1,2).

In this example, the user decides to modify the next class pair of logic node 3.

Due to the limitation of visible space on the display screen there is no interactive 'SHORT' and 'LONG' prompts for the boundary input (i.e., the OLPARS program help function is not available during graphics input).

Session with LONG prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3

ENTER THE LOGIC NODE YOU WISH TO MODIFY - /3/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS

-OR- '*' FOR NEXT CLASS PAIR

-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /*/

(The projected vectors are displayed and the user enters a boundary and a convex point - see short prompt.)

OPTIMLMOD (continued)

ENTER THE CLASS SYMBOL
ASSOCIATED WITH THE CONVEX REGION - /c/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS
-OR- '*' FOR NEXT CLASS PAIR
-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /,/

(The program puts up the menu)

COMMAND NAME: PROJMN

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

After a one-space or two-space display is created, PROJMN displays the mean vectors for the classes displayed superimposed on the screen. A rectangle is drawn around the projection of each mean vector.

USER INTERACTION: NONE

(NOTES: *****)

If the display code in the DI file is not cluster or scatter, or macro or micro, the message 'THE DISPLAY FILE DOES NOT CONTAIN A ONE- OR TWO-SPACE DISPLAY' is printed at the user's terminal.

*****)

PRTCM

COMMAND NAME: PRTCM

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

PRTCM produces a printout of a confusion matrix stored in the display information file.

USER INTERACTION:

NONE

COMMAND NAME: PRTDS

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

PRTDS consists of eight printout options which allow the user to obtain certain basic statistical information about a data set. There are two additional choices available which allow the user to choose the output format and the size of the data set to be processed. When all user prompts are finished, PRTDS will send the output to the system line printer.

Note, PRTDS can be used in excess measurement mode.

USER INTERACTION:

The user is first prompted for a data tree name for which the information will be gathered. The user then selects one or more data output options. The options are entered by number, in any order, with no separators between them. Once the options have been entered and are all valid, an output format prompt will appear. Exponential format may be used any time the user prefers to have values printed in scientific notation. This option should be used when the values being printed are "too large" (magnitude greater than 10^{*9}) or "too small" (magnitude less than $10^{*(-1)}$). Using exponential format, greater accuracy can be seen for values that are "too small". However, the regular format (floating point) is usually easier to read. If the user does not want to use the entire data set, a subset of the data set may be selected.

The effects of the options are as follows:

Option 1: ALL VECTORS

All vectors of the selected data classes are printed. The format of this printout simplifies comparison of the different values of a specific measurement or feature. The tree must contain at least one lowest node for this option to execute.

PRTDS (continued)

Option 2: SINGLE VECTOR

The program will ask for a class name, then for the vector ID number of the vector to be printed. The program then asks if more single vectors are to be selected from this class. If you wish to switch classes, a zero (0) is entered, and the class name prompt will appear again. The standard OLPARS escape (<CR>), entered to any of the prompts in this option will end execution of just this option. This option only works for lowest nodes.

Option 3: RANGES AND OVERLAP

This option prints a table, for each data class in the selected data set, containing the minimum, maximum, and range values for each measurement. When tables for all selected classes have been completed, a table containing the overall minimum, maximum, and range values is printed. Following the tables, an overlap graph is printed containing a line for each class along each measurement. The numeric range for each class will accompany the graph. If just one class has been selected, the overall ranges and overlap graph will not be printed. This option only works for lowest nodes.

Option 4: MEANS AND STANDARD DEVIATIONS

The mean vectors and standard deviations for all data nodes in the selected data set are printed.

Option 5: DIFFERENCE BETWEEN MEAN VECTORS

The absolute values of the difference between each measurement of the mean vectors are printed, along with the Euclidean distance between each pair of mean vectors in the selected data set. This option will not execute if the data set is in excess measurement mode.

Option 6: COVARIANCE MATRICES

The covariance matrix for all data nodes in the selected data set is printed. This option will not execute if the data set is in excess measurement mode.

Option 7: CORRELATION MATRICES

The correlation matrix for all data nodes in the selected data set is printed. This option will not execute if the data set is in excess measurement mode.

Note:

$$\text{Cor}(i,j) = \frac{\text{Cov}(i,j)}{\sqrt{\text{Var}(i) * \text{Var}(j)}}$$

where, $\text{Cor}(i,j)$ = correlation matrix being calculated

$\text{Cov}(i,j)$ = covariance matrix of measurements i and j

$\text{Var}(i)$ = variance of measurement i

$\text{Var}(j)$ = variance of measurement j

Option 8: TREE STRUCTURE

The tree structure of the selected data set is printed in outline form including the dimension of the data set, the number of vectors at each node, the number of nodes in the tree, and the number of lowest nodes in the tree.

EXAMPLE(S):

In the following example(s), `nasal` is a data tree, with `whea`, `rye`, `soys`, and `Clov` all data classes under the senior node.

Session with SHORT prompts

TREENAME? /NASA1/

1. ALL VECTORS
2. SINGLE VECTOR
3. RANGES AND OVERLAP
4. MEANS AND STANDARD DEVIATIONS
5. DIFFERENCE BETWEEN MEAN VECTORS
6. COVARIANCE MATRICES
7. CORRELATION MATRICES
8. TREE STRUCTURE

OPTIONS DESIRED - /138/

EXPONENTIAL FORMAT (Y/N)? /N/

PRINT ENTIRE DATA SET (Y/N)? /Y/

PRTDS (continued)

(PROMPT NOTES: *****)

If option 2, or option 8 are selected individually,
the 'PRINT ENTIRE DATA SET' prompt will not occur.
The 'EXPONENTIAL FORMAT' prompt will also not occur
if option 8 is selected by itself.

*****)

Session with LONG prompts

ENTER THE NAME OF THE DATA SET FOR WHICH
YOU WANT STATISTICAL INFORMATION (8 CHARS. MAX) -
/NASA1/

1. ALL VECTORS
2. SINGLE VECTOR
3. RANGES AND OVERLAP
4. MEANS AND STANDARD DEVIATIONS
5. DIFFERENCE BETWEEN MEAN VECTORS
6. COVARIANCE MATRICES
7. CORRELATION MATRICES
8. TREE STRUCTURE

ENTER THE OPTIONS TO BE PRINTED OUT.
ENTER BY NUMBER, WITH NO SEPARATORS. (E.G. 157) -/2/
DEFAULT OUTPUT FORMAT IS IN FLOATING POINT NOTATION.
WOULD YOU LIKE THE PRINTOUT TO BE IN SCIENTIFIC
NOTATION (Y/N)? /N/
ENTER THE CLASS NAME OF THE VECTOR(S)
TO BE PRINTED - /alfa/

TYPE ZERO (0) TO ENTER ANOTHER CLASS
ENTER THE VECTOR ID NUMBER OF THE VECTOR
TO BE PRINTED OUT - /82/
ENTER THE VECTOR ID NUMBER OF THE VECTOR
TO BE PRINTED OUT - /<CR>/

COMMAND NAME: PRTIDX

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

PRTIDX will identify a class symbol found on the current two-space display or one space micro plot display. Either vector identifiers or the number of vectors within a certain area will be displayed to the user.

The PRTIDX output obtained from a:

1. One space MICRO plot contains the number of vectors, along with the percentage of vectors from each class found in the user requested bin.
2. Two space SCATTER plot contains the identifiers of the vectors found within the user specified region.
3. Two space CLUSTER plot contains the identifiers of the vectors (only upon special request) and a count of the number of vectors found in a single cluster grid location.

USER INTERACTION:

The user is asked:

1. If the output is to be placed in a file or displayed at the terminal.
2. If output is sent to a file, the user is asked for a file name and whether or not the file is to be printed at a line printer.
3. for vectors to be identified or counted.

EXAMPLE(S):

Session with SHORT prompts

TERMINAL OR FILE (T/F)? /F/

FILENAME? /GRAINIDX/

PRINT FILE (Y/N)? /Y/

At this point the current OLPARS display is redisplayed. The following prompt depends upon the type of current display present. In each case, however, the prompt will appear to flash at the user's terminal in the lower left hand portion of the screen. After the prompt is displayed, the graphics cursor will appear for the user to position over the areas to be examined. The following usage of '....' represents user movement of the graphics cursor.

```
MICRO PLOT      - a flashing 'PICK BIN:'  /....B/
SCATTER PLOT    - a flashing 'PICK
                  VECTORS:'    /....L....U/
CLUSTER PLOT    - a flashing 'PICK GRID
                  POSITION:'    /....I/
```

(PROMPT NOTES: *****)

In the previous example, if the user had specified 'T' for the 'TERMINAL OF FILE (T/F)?' prompt, the 'file name' and 'print file' prompts would not have appeared. Note also, if the user terminal is the designated output unit, 1) the OLPARS display is erased and redisplayed after each individual user 'id' request, and 2) PRTIDX queries the user for continuation of the program (CONTINUE (Y/N)?) after each dump of vector ids. (scatter plot indexing has a multiple page facility and uses a 'MORE (Y/N)?' prompt along with the 'continue' prompt).

To exit the program while a graphics cursor is present, the user should type 'Q', for quit.

With a MICRO PLOT display, the user moves the cursor to the bin from which information is desired and enters a space or letter.

With a SCATTER PLOT display, the user is going to request a rectangular region from which to obtain vector identifiers. The user region is defined by two points which represent opposite corners of a rectangle. The user moves the graphics cursor to a point representing one corner of the rectangle and enters a space or letter. Then the user moves the graphics cursor to a point representing the opposite corner and again enters a space or letter. PRTIDX will draw a rectangle using the given points and ask the user if the boundary is acceptable. If the user answers 'No', the graphics cursor will appear again

for the user to re-establish the 'indexing' region. In the given example, the letter 'L' meant lower left corner while 'U' meant upper right corner.

With a CLUSTER PLOT display, the user moves the graphics cursor to the center of the character present in a given grid position and enters a letter or space. If 'I' was the letter entered, then the vector identifiers will appear in the output with the number of vectors from each class found in the given grid position. Otherwise, only the number of vectors from each class will appear in the output.

*****)

Session with LONG prompts

DO YOU WANT THE OUTPUT TO APPEAR AT THE TERMINAL
OR IN A FILE (T/F)? /F/

TYPE IN NAME OF FILE TO BE USED - /GRAINS/

DO YOU WANT FILE PRINTED UPON PROGRAM
COMPLETION (Y/N)? /N/

From this point, the remaining prompts appear in the same manner as shown in the SHORT prompt section.

PRTLOG

COMMAND NAME: PRTLOG

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

PRTLOG gives the user a printer listing of information for any OLPARS logic tree. When all user prompts are finished, PRTLOG will send the output to the system line printer.

USER INTERACTION:

The user is asked for the name of the logic tree to be printed and if (s)he wants statistics to be printed in exponential format.

EXAMPLE(S):

Session with SHORT prompts

LOGIC TREE NAME? /TESTLOG/
EXPONENTIAL FORMAT (Y/N)? /N/

(example of an output listing)

LOGIC TREE NAME IS TESTLOG
DESIGN DATA SET NAME IS TEST (****)
NUMBER OF DIMENSIONS = 4
TOTAL NUMBER OF NODES IN THE LOGIC TREE = 6
NUMBER OF DATA CLASS NODES = 4
NUMBER OF INCOMPLETE LOGIC NODES = 0

| LOGIC NODE | LOGIC CREATION COMMAND (CLASS(ES) PRESENT) |
|------------|--|
| 1 | NMV(ADCS) |
| 6 |**** |
| 5 |ABC |
| 4 |DEFG |
| 3 |CAT |
| 2 |SAND |

At this point statistics are printed for each logic node. Output varies depending on the type of logic at the node:

PAIRWISE LOGIC

MINIMUM VOTE COUNT.

for each class pair:

LOGIC TYPE.

OPTION CREATING THE LOGIC.

if logic type is FISHER:

FISHER COEFFICIENTS.

DISCRIMINANT COEFFICIENTS.

THRESHOLDS AVAILABLE.

NUMBER OF THRESHOLDS USED AND WHICH ARE USED (1-5).

CLASS ON NUMERICALLY GREATER SIDE OF MIDDLE

THRESHOLD.

if logic type is OPTIMAL DISCRIMINANT PLANE:

NUMBER OF LINE SEGMENTS IN THE BOUNDARY.

DISCRIMINANT COEFFICIENT OF EACH LINE SEGMENT.

THRESHOLD FOR EACH LINE SEGMENT.

CLASS ON CONVEX SIDE OF THE BOUNDARY.

CLASS IN EXCESS REGION.

GROUP LOGIC

LOGIC TYPE FOR GROUP (one-space, two-space,
or Boolean).

OPTION CREATING THE LOGIC.

for one-space logic:

DISCRIMINANT COEFFICIENTS OF PROJECTION VECTOR.

THRESHOLD(\$).

for two-space logic:

NUMBER OF LINE SEGMENTS IN BOUNDARY(IES).

DISCRIMINANT COEFFICIENTS FOR EACH LINE SEGMENT.

THRESHOLD FOR EACH LINE SEGMENT.

LOGIC NODE NUMBERS ASSOCIATED WITH EACH REGION.

NEAREST MEAN VECTOR LOGIC

OPTION CREATING THE LOGIC.
TYPE OF WEIGHTING (OR QUADRATIC CLASSIFIER).
MEASUREMENTS IGNORED.
REJECT BOUNDARY DISTANCE VALUES.
for each class at the logic node:
 MEAN VECTOR.
 WEIGHTING VECTOR.
 WEIGHTING MATRIX.
 DETERMINANT OF THE COVARIANCE MATRIX FOR THE CLASS.
 LOGIC NODE NUMBER ASSOCIATED WITH THE CLASS.

NEAREST NEIGHBOR LOGIC

OPTION CREATING THE LOGIC.
NUMBER OF NEAREST NEIGHBORS EXAMINED.
MEASUREMENTS IGNORED.
TREE NAME OF THE REFERENCE PATTERNS.
LOGIC NODE NUMBERS ASSOCIATED WITH EACH CLASS AT THE
NODE.

(PROMPT NOTES: *****)

A 'Y' (yes) response to the prompt 'EXPONENTIAL FORMAT (Y/N)?' causes statistics to be printed in scientific notation. Any other response causes statistics to be printed in floating point format. Exponential format should be used when the values being printed are "too large" (magnitude greater than 10^{**9}) or "too small" (magnitude less than $10^{*(-1)}$). Using exponential format, greater accuracy can be seen for values that are "too small". However, floating point notation is usually easier to read.

*****)

Session with LONG prompts

ENTER THE NAME OF THE LOGIC TREE TO BE PRINTED
(* REPRESENTS THE CURRENT LOGIC) - /TESTLOG/
DEFAULT OUTPUT FORMAT IS IN FLOATING POINT NOTATION.
WOULD YOU LIKE THE PRINTOUT TO BE IN SCIENTIFIC
NOTATION (Y/N)? /N/

(output is same as in short prompt notes.)

COMMAND NAME: PWEVAL

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

PWEVAL evaluates pairwise logic at a user chosen logic node. (See FISHER for additional information).

This command can only be used after FISHER has created logic in the current logic tree, or FISHMOD, THRESHMOD, or OPTIMLMOD has modified existing pairwise logic.

NOTE, If this command is used twice at the same logic node, without deleting the existing logic, a confusion matrix of zeroes will be created.

USER INTERACTION:

The user is asked which pairwise logic node should be evaluated.

EXAMPLE(S):

In the following example, the user has decided to evaluate vectors at pairwise logic node number 3.

Session with SHORT prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3
NODE NUMBER? /3/

(Confusion matrix is displayed.)
(The program puts up the menu)

In this example the user decides to evaluate vectors at logic node 3.

Session with LONG prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3
ENTER THE LOGIC NODE YOU WISH TO EVALUATE - /3/

(Confusion matrix is displayed.)
(The program puts up the menu)

RANK

COMMAND NAME: RANK

CATEGORY: MEASUREMENT EVALUATION COMMAND (subsidiary)

FUNCTIONAL DESCRIPTION:

RANK displays the type of measurement evaluation ranking specified by the user. In an 'overall' ranking, the discriminant measurements displayed are those values which separate all classes on the basis of each measurement. An overall ranking also shows the class that is best separated from all other classes ("best class") by each measurement, and the two classes that are best separated from each other ("best class pair") by each measurement. In a 'measurement by class' ranking, the discriminant measurements displayed are those values which separate the user-specified class from all other classes. In a 'measurement by class pair' ranking, the discriminant measurements displayed are those values which separate the first user-specified class from the second user-specified class. In the three types of rankings described above, the measurement numbers which correspond to either the smallest or largest discriminant values are the best measurements for the type of separation being performed.

In a 'class by measurement' ranking, the user specifies a measurement number which is used as an index into each discriminant measurement vector whose values separate one class from all other classes (ex. class i separated from all other classes). There are as many vectors as there are classes. The discriminant measurement values which correspond to the user-specified measurement are ranked, and the results show which classes are best separated from all other classes using the user-specified measurement.

In a 'class pair by measurement' ranking, the user specifies a measurement number which is used as an index into each discriminant measurement vector whose values separate one class from another class (ex. class i separated from class j). There are as many discriminant measurement vectors as there are combinations (in twos) of classes ("class pairs"). (NOTE: A separation between class i and class j is the same as a separation between class j and class i (the order of class symbols within a class pair is insignificant), and thus constitutes one combination (class pair). Also, a class is never evaluated to be separated from itself, that is, class i separated from class i is never a class pair). The discriminant measurement values which correspond to the user-specified measurement are ranked and the results show which two classes are best separated from each other using the

user-specified measurement.

For each type of ranking, all discriminant measurement values are ranked in ascending (PROBCONF) or descending (DSCRMEAS) order, depending on the 'CURRENT OPTION' (most recently executed Major Measurement Evaluation Command). Therefore, those values at the top of the list are the best values for the type of ranking being performed.

Note, RANK can be used in excess measurement mode.

USER INTERACTION:

The user is asked to select one of the following types of rankings:

- (1) overall
- (2) measurement by class
- (3) measurement by class pair
- (4) class by measurement
- (5) class pair by measurement

If the user selects a 'measurement by class' ranking, s(he) is asked to enter the class symbol which represents the class for which measurements will be ranked. If the user selects a 'measurement by class pair' ranking, s(he) is asked to enter two class symbols representing the pair of classes for which measurements will be ranked. If the user selects a 'class by measurement' or a 'class pair by measurement' ranking, s(he) is asked for the measurement number for which classes or class pairs will be ranked.

RANK (continued)

EXAMPLE(S):

Session with SHORT prompts

In the following example the user has chosen to rank measurements by class pair s/C

RANKING OPTIONS:

(1) OVERALL
(2) MEAS. BY CLASS
(3) MEAS. BY CLASS PAIR
(4) CLASS BY MEAS.
(5) CLASS PAIR BY MEAS.
TYPE OF RANKING = /3/
FIRST CLASS SYMBOL: /s/
SECOND CLASS SYMBOL: /C/

RANKING OPTIONS:

(1) OVERALL
(2) MEAS. BY CLASS
(3) MEAS. BY CLASS PAIR
(4) CLASS BY MEAS.
(5) CLASS PAIR BY MEAS.
TYPE OF RANKING = /<CR>/

(PROMPT NOTES: *****)

RANK remains in prompt mode until the user selects a valid option number and when applicable a valid class symbol or a valid measurement number, or until the OLPARS character for exiting commands is typed. In the case of a 'measurement by class pair' ranking, the second class symbol typed must be different from the first class symbol typed.

*****)

Session with LONG prompts

In the following example, the user has chosen to rank all class pairs on the basis of measurement 4.

RANKING OPTIONS:

(1) OVERALL
(2) MEASUREMENT BY CLASS
(3) MEASUREMENT BY CLASS PAIR
(4) CLASS BY MEASUREMENT
(5) CLASS PAIR BY MEASUREMENT
TYPE IN THE NUMBER (1 THRU 5)
THAT CORRESPONDS TO THE TYPE OF RANKING DESIRED: /5/
TYPE IN THE MEASUREMENT NUMBER TO BE USED
MEAS NO. = /4/

RANKING OPTIONS:

(1) OVERALL
(2) MEASUREMENT BY CLASS
(3) MEASUREMENT BY CLASS PAIR
(4) CLASS BY MEASUREMENT
(5) CLASS PAIR BY MEASUREMENT
TYPE IN THE NUMBER (1 THRU 5)
THAT CORRESPONDS TO THE TYPE OF RANKING DESIRED: /<CR>/

RDISPLAY

COMMAND NAME: RDISPLAY

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

RDISPLAY will redisplay the previous one or two space data projection or the previous confusion matrix at the user's terminal.

USER INTERACTION:

NONE

COMMAND NAME: REASNAME

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

Logic Design Commands initialize the reassociated class names in a logic tree to be the same as the original design data set class names. REASNAME gives the user the ability to change these reassociated class names to whatever is desired. It is useful to change the reassociated class names in cases where a test data set is to be evaluated against logic designed on a data set whose class names are different from the test data set. In such a case, the reassociated class names in the current logic should be changed to be the same as the class names in the test data set, and the user should use the reassociated class names during overall logic evaluation.

REASNAME displays, for the current logic, a table of completed logic nodes, along with the original design data set class names and the reassociated class names. In order to use REASNAME, the current logic must be complete.

USER INTERACTION:

The user is prompted for a logic node number and a new reassociated class name until (s)he is satisfied with the changes made.

EXAMPLE(S):

Session with SHORT prompts

In the following example, the user changes the reassociated class names for all of the logic nodes. When the table is redisplayed, (s)he discovers that (s)he has made a spelling error ('oots' should be 'oats'). (S)He responds with a 'Y' (yes) to the 'MORE CHANGES (Y/N)? ' prompt and corrects the error. (S)He is then satisfied with the results and responds with a 'N' (no) to the 'MORE CHANGES (Y/N)? ' prompt.

REASNAME (continued)

| LOGIC NODE | DESIGN DATA SET CLASS NAME | REASSOCIATED CLASS NAME |
|------------|-------------------------------|----------------------------|
| ----- | ----- | ----- |
| 5 | anod | anod |
| 6 | bnod | bnod |
| 8 | cnod | cnod |
| 9 | dnod | dnod |

LOGIC NODE AND CLASS NAME: /5 oots/
 LOGIC NODE AND CLASS NAME: /6 weat/
 LOGIC NODE AND CLASS NAME: /8 soys/
 LOGIC NODE AND CLASS NAME: /9 alfa/
 LOGIC NODE AND CLASS NAME: /0/

| LOGIC NODE | DESIGN DATA SET CLASS NAME | REASSOCIATED CLASS NAME |
|------------|-------------------------------|----------------------------|
| ----- | ----- | ----- |
| 5 | anod | oots |
| 6 | bnod | weat |
| 8 | cnod | soys |
| 9 | dnod | alfa |

MORE CHANGES (Y/N)? /Y/
 LOGIC NODE AND CLASS NAME: /5 oats/
 LOGIC NODE AND CLASS NAME: /0/

| LOGIC NODE | DESIGN DATA SET CLASS NAME | REASSOCIATED CLASS NAME |
|------------|-------------------------------|----------------------------|
| ----- | ----- | ----- |
| 5 | anod | oats |
| 6 | bnod | weat |
| 8 | cnod | soys |
| 9 | dnod | alfa |

MORE CHANGES (Y/N)? /N/

(PROMPT NOTES: *****)

If the current logic is incomplete, REASNAME notifies the user and exits. If the user is running OLPARS in long prompt mode, (s)he is given some instructions and asked if (s)he wants to continue. If the user wants to continue, or if (s)he is running OLPARS in short prompt mode, the table of logic nodes is displayed. Pages of the logic node table are displayed until the entire table has been shown, or until the user does not respond with 'Y' (yes) to the 'NEXT PAGE OF LOGIC NODES (Y/N)? ' prompt. Next, for each logic node in the table whose reassocated name

REASNAME (continued)

is to be changed, the user is asked to enter the logic node number and the new reassociated class name. The user is notified if a logic node number is invalid, and prompting continues. If at any time the user types zero (0) for the logic node number, the table of logic nodes is redisplayed, with modifications, and the user is asked if (s)he wants to make any more changes. Prompting for a logic node number and class name will continue until the user exits via the OLPARS program exit convention or until a response indicates (s)he is satisfied.

*****)

Session with LONG prompts

A TABLE OF LOGIC NODE NUMBERS, DESIGN DATA SET CLASS NAMES, AND REASSOCIATED CLASS NAMES WILL BE DISPLAYED. YOU MAY CHANGE THE REASSOCIATED CLASS NAME OF ANY LOGIC NODE IN THE TABLE BY TYPING THE LOGIC NODE NUMBER AND THE NEW REASSOCIATED CLASS NAME ON ONE LINE. IF AT ANY TIME YOU WISH TO HAVE THE TABLE REDISPLAYED, TYPE IN A ZERO (0) IN RESPONSE TO THE 'LOGIC NODE NUMBER AND REASSOCIATED CLASS NAME' PROMPT.

DO YOU WISH TO CONTINUE (Y/N)? /Y/

(The table would appear here as in the above example).

LOGIC NODE NUMBER AND REASSOCIATED CLASS NAME: /5 oats/
LOGIC NODE NUMBER AND REASSOCIATED CLASS NAME: /6 weat/
LOGIC NODE NUMBER AND REASSOCIATED CLASS NAME: /8 soys/
LOGIC NODE NUMBER AND REASSOCIATED CLASS NAME: /9 alfa/
LOGIC NODE NUMBER AND REASSOCIATED CLASS NAME: /0/

(The table would be redisplayed here with modifications as in the above example).

DO YOU WISH TO CHANGE MORE REASSOCIATED CLASS NAMES (Y/N)? /N/

REDRAW

COMMAND NAME: REDRAW

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

REDRAW is used to draw an existing one or two-space decision boundary on an OLPARS display.

USER INTERACTION: NONE

(NOTES: *****)

A message is printed at the terminal if the projection is not a one or two-space display. The program will then exit with no action taken.

(If a data partition (boundary) has been drawn on a one or two space display and a new display of the same data projection appears on the screen at some later time, the boundary does not automatically reappear.)

On a two-space projection, REDRAW will extend the line segments to the edge of the display window.

REDRAW also reconstructs boundaries drawn on the original projection onto "zoomed" projections and vice versa.

Two-space boundaries are labeled in order of creation ('+' for first boundary drawn, 'X' for second boundary drawn).

*****)

AD-A118 733

PAR TECHNOLOGY CORP. NEW HARTFORD NY

F/G 9/2

ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. USE--ETC(U)

JUN 82 S E HAEHN; D MORRIS

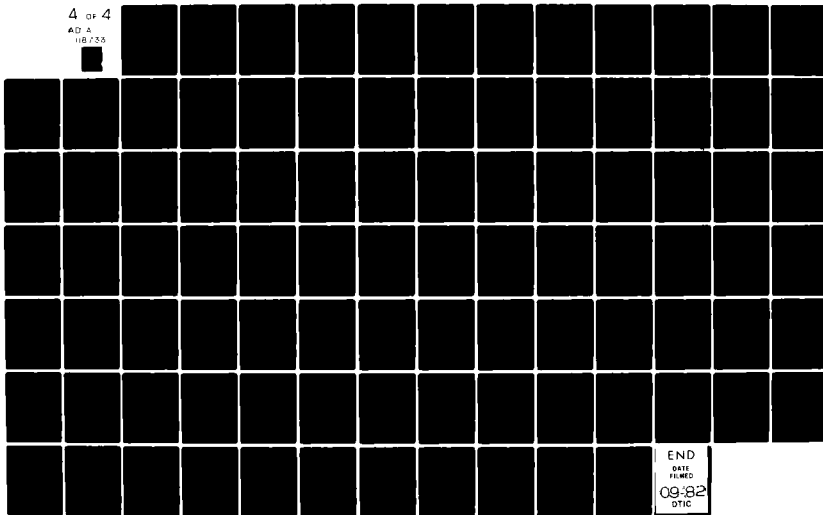
UNCLASSIFIED

PAR-82-21

NL

4 of 4

AD A
118733



END
DATE
FILMED
09-82
DTIC

REPROJECT

COMMAND NAME: REPROJECT

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

REPROJECT enables a user to project a data set on different eigenvectors after an EIGEN VALUE or GENERAL DISCRIMINANT projection command has created a one-space or two-space display.

USER INTERACTION:

User is asked:

- 1) if a printer listing of the eigen values is desired.
- 2) to select one or two eigen values (depending whether the initial projection was a one space or two space projection) to use in the projection.
- 3) if all the vectors of the classes that lie at a logic node are to be used in subsequent computations, or only those vectors that lie at the logic node.

(NOTE, the above prompt only appears when the previous projection was created by a logic design command.)

EXAMPLE(S):

The following example indicates that the number of measurements used in eigen value calculations was three. The previous projection could be created by S1EIGV, S2EIGV, L1EIGV, L2EIGV, S1GNDV, S2GNDV, L1GNDV, or L2GNDV. In this example the previous projection was created by S2EIGV.

Session with SHORT prompts

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 1.751099E+04 |
| 2 | 1.378148E+03 |
| 3 | 3.736754E+02 |

PRINTOUT? Y/N /Y/

EIGENVECTOR NO. FOR THE X PROJECTION: /1/
EIGENVECTOR NO. FOR THE Y PROJECTION: /3/

REPROJECT (continued)

(PROMPT NOTES: *****)

Eigenvector Number -

The program will continue to prompt until the user types in a valid eigenvector number (greater than zero and less than or equal to the number of measurements used in the computation).

Printout? -

The program stays in prompt mode until the user types 'Y' or 'N' as the first character typed. If the user types 'Y' a lineprinter copy of the above NUMBER/EIGENVALUE table is produced.

We obtained two eigen vector projection prompts because the previous projection was created by a two space projection command. If a one space projection command was used, only the X-projection prompt would appear.

*****)

Session with LONG prompts

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 1.751099E+04 |
| 2 | 1.378148E+03 |
| 3 | 3.736754E+02 |

DO YOU WANT A PRINTOUT OF THE EIGENVALUES?
TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /N/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE X PROJECTION:
/1/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE Y PROJECTION:
/2/

COMMAND NAME: RESTRUCT

CATEGORY: STRUCTURE ANALYSIS COMMAND

FUNCTIONAL DESCRIPTION:

RESTRUCT (restructure data set) uses the boundary(s) on a one-space or two space plot to divide a class of the current data set into two or three subclasses. RESTRUCT presents a list of data classes displayed and asks the user to select a class to be restructured. RESTRUCT then asks the user to type in two (for displays with one boundary) or three (for displays with two boundaries) unique four character node names. Each node name typed is associated with a region on the display. The vectors at the node being restructured are determined to lie in one of the display regions, and are assigned to the corresponding node (subclass) in the restructured data tree. (See Appendix B for an explanation of the display regions and the method of assigning a vector to a region). RESTRUCT checks to be sure that the projection being used has been created by a structure analysis routine.

USER INTERACTION:

The user is asked to:

- 1) Select one class to be restructured.
- 2) Rename the 2 or 3 newly created classes, depending on the number of boundary(s) drawn; 2 classes for one boundary, 3 classes for two boundaries.

EXAMPLE(S):

The following examples assume that ABCD, EFGH, IJKL, and MNOP are the lowest nodes in the current data set. In the example using short prompts, the display is a two-space projection and two boundaries are drawn. Class EFGH will be restructured into the new classes 1111, 2222, and 3333. In the example using long prompts, the display is a one-space projection and one boundary is drawn. Class IJKL will be restructured into the new classes 'qrst' and 'uvwx'.

RESTRUCT (continued)

Session with SHORT prompts

SELECT ONE CLASS TO BE RESTRUCTURED FROM THE FOLLOWING

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /E/

RENAME THE RESTRUCTURED CLASS -

NAME OF FIRST CONVEX REGION? /1111/

NAME OF SECOND CONVEX REGION? /2222/

NAME OF REMAINING REGION? /3333/

(PROMPT NOTES: *****)

Class Symbol -

If the class symbol typed is an invalid class symbol, i.e. it doesn't exist, RESTRUCT will prompt the user again for a valid class symbol. If more than one symbol is typed, the message 'INCORRECT NUMBER OF CLASSES SELECTED. EXACTLY ONE CLASS MUST BE CHOSEN' will be printed, and the user will be prompted again.

Display Symbols -

Because the display symbol of the restructured class will no longer be used in future projections, it may be used as the display symbol of one of the newly created classes. All display symbols used in the projections must be unique, including the newest classes.

When renaming the restructured class, if an illegal class name is entered, the message 'ILLEGAL NODE NAME ENTERED' is printed and the user is prompted again.

If the new name is not unique in the data set, the message 'NON-UNIQUE NODE NAME ENTERED' is printed and the user is prompted again. If just the display symbol is not unique, the message 'NON-UNIQUE DISPLAY SYMBOL ENTERED' is printed and the user is prompted again.

RESTRUCT (continued)

If a new class contains no vectors, the message 'CLASS ---- IS EMPTY' will be printed (where ---- is the name of the class). If there are two new classes being created and either class is empty (contains no vectors) the old class is not restructured. When there are three new classes being created, at least two of the new nodes must be non-empty (contain vectors) in order to restructure the old class.

When a class has successfully been restructured, a message of the form 'DATA CLASS ---- HAS SUCCESSFULLY BEEN DIVIDED INTO THE NEW CLASSES ----, ----, AND ---- ' will be printed.

*****)

Session with LONG prompts

FROM THE LIST OF CLASSES BELOW, TYPE IN THE CLASS SYMBOL OF ONE CLASS TO BE RESTRUCTURED (I. E. S)

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

TYPE CLASS SYMBOLS ON A SINGLE LINE (E.G. XYZW)
A MINUS SIGN AS THE FIRST CHARACTER INDICATES THAT
EVERYTHING BUT THE SYMBOLS TYPED ARE TO BE SELECTED.

WHEN A '*' IS TYPED, ALL CLASS SYMBOLS WILL BE SELECTED.
WHEN USED WITH A MINUS SIGN, NO CLASS SYMBOLS WILL BE
SELECTED.

CLASS SYMBOLS: /I/

RENAME THE CLASS BEING RESTRUCTURED. TYPE IN THE NEW NODE
NAMES OF EACH REGION RESULTING FROM THE BOUNDARY(S) DRAWN

TYPE IN THE NEW NODE NAME (4 CHARACTERS LONG)
FOR THE LEFT REGION OF THE RESTRUCTURED CLASS
(I. E. XZ WV) - /qrst/

TYPE IN THE NEW NODE NAME (4 CHARACTERS LONG)
FOR THE RIGHT REGION OF THE RESTRUCTURED CLASS
(I. E. XZ WV) - /uvw x/

S1CRDV

COMMAND NAME: S1CRDV

CATEGORY: STRUCTURE ANALYSIS COMMAND

FUNCTIONAL DESCRIPTION:

S1CRDV allows a user to project the current data set onto a single coordinate for use in structure analysis. A one-space display is created at the terminal.

Note, S1CRDV can be used in excess measurement mode.

USER INTERACTION:

The user is asked to select one coordinate (measurement number) for use in the projection.

EXAMPLE (S):

Session with SHORT prompts

COORDINATE NO.? /12/

INITIALIZING DISPLAY FILE HEADERS

(PROMPT NOTES: *****)

Coordinate Number -

The coordinate number entered by the user must be greater than zero and less than or equal to the vector dimensionality. If an invalid coordinate number is entered, an error message will occur and the program will continue to prompt the user for a valid coordinate number.

*****)

Session with LONG prompts

ENTER THE MEASUREMENT COORDINATE NUMBER
TO USE FOR THE X PROJECTION - /5/

INITIALIZING DISPLAY FILE HEADERS

COMMAND NAME: S1EIGV

CATEGORY: STRUCTURE ANALYSIS COMMAND

FUNCTIONAL DESCRIPTION:

S1EIGV projects a data set onto an eigenvector for use in structure analysis. A brief description of the program algorithm follows:

- 1) The user is given the option to eliminate any measurements from the projection.
- 2) The covariance matrix of the current data set is used to compute the eigenvalues and eigenvectors.
- 3) The eigenvalues will be displayed in descending order at the user's terminal. The user will select one eigenvalue number. The eigenvector which corresponds to the eigenvalue number selected by the user will be used as a projection vector.
- 4) For each class in the current data set, the data vectors will be projected onto the projection vector, i.e.,

$X(i)$ = the projection of data vector i onto the projection vector

- 5) A macro plot or micro plot will be displayed at the user's terminal.

USER INTERACTION:

The user is asked:

- 1) if any measurements are to be eliminated from the computation of the projection vector
- 2) if he wants a printout of the eigenvalues
- 3) to select one eigenvalue for use in the projection

S1EIGV (continued)

EXAMPLE (S):

The following examples assume a vector dimensionality equal to 4.

Session with SHORT prompts

INITIALIZING DISPLAY FILE HEADERS

ELIMINATE MEASUREMENTS? Y/N /Y/

MEAS = /2/
MEAS = /<CR>/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 1.751099E+04 |
| 2 | 1.378148E+03 |
| 3 | 3.736754E+02 |

PRINTOUT? Y/N /Y/

EIGENVECTOR NO. FOR THE X PROJECTION: /1/

(PROMPT NOTES: *****)

Measurement Number -

If the user types a carriage return the first time this prompt occurs, the program will exit. The following checks are made on the measurement number:

- 1) check for a valid measurement number (greater than zero and less than or equal to the vector dimensionality)
- 2) check to see that the user does not type the same measurement number twice
- 3) check to see that the user does not eliminate all the measurements

Note: errors from conditions 1 and 2 above cause the user to remain in prompt mode; an error from condition 3 causes an error message to be printed and the program to exit.

Eigenvector Number -

The program stays in prompt mode until the user types in a valid eigenvector number (greater than zero and less than or equal to the number of measurements used in the computation)

Printout? -

The program stays in prompt mode until the user types Y or N as the first character typed. If the user types Y a lineprinter copy of the above NUMBER/EIGENVALUE table is produced

*****)

Session with LONG prompts

INITIALIZING DISPLAY FILE HEADERS

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE COMPUTATION?

TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /Y/

MEASUREMENT NUMBER = /1/

MEASUREMENT NUMBER = /3/

MEASUREMENT NUMBER = /<CR>/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 3.143891E+04 |
| 2 | 2.442550E+03 |

DO YOU WANT A PRINTOUT OF THE EIGENVALUES?

TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /N/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE X PROJECTION:

/1/

S2CRDV

COMMAND NAME: S2CRDV

CATEGORY: STRUCTURE ANALYSIS COMMAND

FUNCTIONAL DESCRIPTION:

S2CRDV allows a user to project the current data set onto two coordinates for use in structure analysis. A two-space display is created at the terminal.

Note, S2CRDV can be used in excess measurement mode.

USER INTERACTION:

The user is asked to select two coordinates (measurement numbers) to be used in the projection.

EXAMPLE(S):

Session with SHORT prompts

COORDINATE NO. FOR THE X PROJECTION: /1/

COORDINATE NO. FOR THE Y PROJECTION: /2/

INITIALIZING DISPLAY FILE HEADERS

(PROMPT NOTES: *****)

Coordinate Numbers -

The coordinate numbers entered by the user must be greater than zero and less than or equal to the vector dimensionality. If an invalid coordinate number is entered, an error message will occur and the program will continue to prompt the user for a valid coordinate number.

*****)

Session with LONG prompts

ENTER THE COORDINATE NUMBER TO USE FOR THE X PROJECTION:
/2/

ENTER THE COORDINATE NUMBER TO USE FOR THE Y PROJECTION:
/5/

INITIALIZING DISPLAY FILE HEADERS

COMMAND NAME: S2EIGV

CATEGORY: STRUCTURE ANALYSIS COMMAND

FUNCTIONAL DESCRIPTION:

S2EIGV projects a data set onto a pair of eigenvectors for use in structure analysis. A brief description of the program algorithm follows:

- 1) The user is given the option to eliminate any measurements from the projection.
- 2) The covariance matrix of the current data set is used to compute the eigenvalues and eigenvectors.
- 3) The eigenvalues will be displayed at the user's terminal (in descending order). The user will select two eigenvalue numbers. The two eigenvectors which correspond to the two eigenvalue numbers selected by the user will be used as projection vectors.
- 4) For each class in the current data set, the data vectors will be projected onto the projection vector, i.e.,
$$X(i) = \text{the projection of data vector } i \text{ onto projection vector 1}$$
$$Y(i) = \text{the projection of data vector } i \text{ onto projection vector 2}$$
- 5) a scatter plot or a cluster plot will be displayed at the user's terminal.

USER INTERACTION:

The user is asked:

- 1) if any measurements are to be eliminated from the computation of the projection vectors
- 2) if he wants a printout of the eigenvalues
- 3) to select two eigenvalues for use in the projection

S2EIGV (continued)

EXAMPLE(S):

The following examples assume a vector dimensionality equal to 4.

Session with SHORT prompts

INITIALIZING DISPLAY FILE HEADERS

ELIMINATE MEASUREMENTS? Y/N /Y/

MEAS # = /2/

MEAS # = /<CR>/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 1.751099E+04 |
| 2 | 1.378148E+03 |
| 3 | 3.736754E+02 |

PRINTOUT? Y/N /Y/

EIGENVECTOR NO. FOR THE X PROJECTION: /1/

EIGENVECTOR NO. FOR THE Y PROJECTION: /2/

(PROMPT NOTES: *****)

Measurement Number -

If the user types a carriage return the first time this prompt occurs, the program will exit. The following checks are made on the measurement number:

- 1) check for a valid measurement number (greater than zero and less than or equal to the vector dimensionality)
- 2) check to see that the user does not type the same measurement number twice
- 3) check to see that the user does not eliminate all the measurements

Note: errors from conditions 1 and 2 above cause the user to remain in prompt mode; an error from condition 3 causes an error message to be printed and the program to exit.

Eigenvector Number -

The program stays in prompt mode until the user types in a valid eigenvector number (greater than zero and less than or equal to the number of measurements used in the computation)

Printout? -

The program stays in prompt mode until the user types Y or N as the first character typed. If the user types Y a lineprinter copy of the above NUMBER/EIGENVALUE table is produced

***** !*****)

Session with LONG prompts

INITIALIZING DISPLAY FILE HEADERS

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE COMPUTATION?

TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /Y/

MEASUREMENT NUMBER = /1/

MEASUREMENT NUMBER = /3/

MEASUREMENT NUMBER = /<CR>/

| NUMBER | EIGENVALUE |
|--------|--------------|
| 1 | 3.143891E+04 |
| 2 | 2.442550E+03 |

DO YOU WANT A PRINTOUT OF THE EIGENVALUES?

TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /N/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE X PROJECTION: /1/

ENTER THE EIGENVECTOR NUMBER TO USE FOR THE Y PROJECTION: /2/

S2FSHP

COMMAND NAME: S2FSHP

CATEGORY: STRUCTURE ANALYSIS COMMAND

FUNCTIONAL DESCRIPTION:

S2FSHP projects the selected data set on two fisher directions which coorespond to two pairs of data classes within the selected data set. Note, the resultant fisher discriminant vectors are orthonormalized before they are used for data projection.

USER INTERACTION:

The user is asked:

- 1) To select a pair of classes to be used in the computation of the first fisher discriminant.
- 2) To select a pair of classes to be used in the computation of the second fisher discriminant.
- 3) If covariance or scatter matrices are to be used in the computations.
- 4) If any measurements are to be eliminated from the computations.

EXAMPLE(S):

The following examples assume that ABCD,EFGH,IJKL, and MNOP are classes in the data set. Classes ABCD and MNOP will be used to compute fisher discriminant 1 and EFGH and MNOP will be used to compute fisher discriminant 2. Scatter matrices will be used in the computations, and measurement number 3 will be eliminated.

Session with SHORT prompts

INITIALIZING DISPLAY FILE HEADERS

SELECT TWO CLASSES FROM THE LIST BELOW TO USE IN
THE COMPUTATION OF FISHER 1

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /AM/

SELECT TWO CLASSES FROM THE LIST BELOW TO USE IN
THE COMPUTATION OF FISHER 2

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /EM/

COVARIANCE/SCATTER MATRIX OPTION? 0/1 /1/

ELIMINATE MEASUREMENTS? Y/N /Y/

MEAS = /3/

MEAS = /<CR>/

(PROMPT NOTES: *****)

If the two fisher discriminants are linearly dependent,
the message 'THE TWO FISHER PROJECTION VECTORS ARE
LINEARLY DEPENDENT. THE X AND Y PROJECTION VECTORS WILL
BE THE SAME VECTOR.' is printed.

Class Symbols -

The class symbol string must be typed on a single
line with no intermittent blanks, commas, and the
like. If the class symbols typed are invalid class
symbols, i.e. they don't exist, S2FSHP will prompt
the user again for some valid class symbols. If
the user types in one or more invalid class symbols,
then all valid class symbols must be retyped on the
next try.

*****)

S2FSHP (continued)

Session with LONG prompts

INITIALIZING DISPLAY FILE HEADERS

FROM THE LIST OF LOWEST NODES BELOW, SELECT THE
FIRST AND SECOND CLASS TO BE USED TO COMPUTE
FISHER 1 (IE. CS)

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /AM/
FROM THE LIST OF LOWEST NODES BELOW, SELECT THE
FIRST AND SECOND CLASS TO BE USED TO COMPUTE
FISHER 2 (IE. CS)

CLASS SELECT LIST

ABCD EFGH IJKL MNOP

CLASS SYMBOLS: /EM/

ARE COVARIANCE OR SCATTER MATRICES TO BE USED IN THE
COMPUTATION OF THE FISHER DISCRIMINANT
(0-COVARIANCE, 1-SCATTER)? /1/

DO YOU WISH TO ELIMINATE MEASUREMENTS FROM THE
COMPUTATION?
TYPE Y FOR YES, N FOR NO, <CR> TO EXIT /Y/

MEASUREMENT NUMBER = /3/
MEASUREMENT NUMBER = /<CR>/

SCALRET

COMMAND NAME: SCALRET

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

SCALRET is the complementary function of SCALZM. Upon completion of one or more zooming operations on a one or two-space display, SCALRET allows you to obtain the original global display.

USER INTERACTION: NONE

SCALZM

COMMAND NAME: SCALZM

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

SCALZM (scale zoom) changes the scale of a one-space or two-space display by "zooming in" on a user defined subset of the current display.

USER INTERACTION:

FOR ONE SPACE DISPLAYS ONLY

Using the graphics cursor, you are to select new minimum and maximum points for which the display is to be expanded. The first time, position the graphics cursor at the left-most point on the baseline of the display and enter any character except 'Q'. The second time, position the graphics cursor at the right-most point on the baseline and again enter any character except 'Q'. (If a 'Q' is entered, you will immediately exit SCALZM.) Two lines are drawn on the current display indicating the area to be zoomed. You will be asked if the border is correct. If the reply is yes ('Y'), then SCALZM will continue. If the reply is no ('N'), then you will be able to zoom another subarea of the current display

FOR TWO SPACE DISPLAYS ONLY

Using the graphics cursor, you are to select the portion of the current display which is to be expanded. The first time, position the graphics cursor at the lower left hand corner of the desired display area and enter any character except 'Q'. The second time position the graphics cursor at the upper right hand corner of the desired display area and again enter any character except 'Q'. (If a 'Q' is entered, you will immediately exit SCALZM.) A box will be drawn on the current display indicating the area to be zoomed. You will be asked if the border (box) is correct. If the reply is yes ('Y'), then SCALZM will continue. If the reply is no ('N'), then you will be able to zoom another subarea of the current display.

SCALZM (continued)

(NOTES: *****)

Due to the limitations of visible space on the display screen, there will not be any differences between 'SHORT' and 'LONG' prompts. If you enter a '?', it will be accepted as an input for the new subarea.

However, if you have changed the default prompt mode to be 'long' by means of the 'CDEFAULT' command, you will receive a preliminary explanation of how to respond to the prompts issued by SCALZM.

*****)

SELECT

COMMAND NAME: SELECT

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

SELECT gives the user the ability to determine which class symbols are to be displayed at the OLPARS user's terminal in a one- or two-space display. SELECT checks the Display Information (DI) File for a valid display code, cluster or scatter, or macro or micro. If the display code is valid, the user is asked to type in the class symbols to be displayed. Otherwise, the program exits. SELECT writes the display flag for each DI File logical entry. If the user has indicated that a class symbol is to be displayed, the display flag for that class is "turned on" (set). Otherwise, the display flag for the given class is "turned off" (not set). The appropriate display (cluster, scatter, macro, or micro) is then put up at the user's terminal.

USER INTERACTION:

The user is asked to type in the class symbols.

EXAMPLE(S):

The following two examples assume that class symbols A,B,C,D,E,F,G, and H exist in the DI file.

Session with SHORT prompts

For the following example class symbols A,B,C, and D will be displayed. Class symbols E,F,G, and H will not be displayed.

CLASS SYMBOLS: /ABCD/

SELECT (continued)

(PROMPT NOTES: *****)

Class Symbols -

The class symbol string must be typed on a single line with no intermittent blanks, commas, and the like. If the class symbols typed do not exist in the DI file (invalid class symbols), SELECT will stay in prompt mode. If the user types in one or more invalid class symbols, then all valid class symbols must be retyped on the next try. A minus sign preceding the class symbols indicates that the class symbols typed are not to be displayed. Typing only a minus sign is invalid and SELECT remains in prompt mode.

*****)

Session with LONG prompts

For the following example, class symbols B,D,F, and H will be displayed. Class symbols A,C,E, and G will not be displayed.

TYPE CLASS SYMBOLS ON A SINGLE LINE. (e.g. ABCD).
A MINUS SIGN AT THE BEGINNING INDICATES THAT THE CLASS
SYMBOLS ARE NOT TO BE DISPLAYED.
CLASS SYMBOLS: /-ACEG/

SETDS

COMMAND NAME: SETDS

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

SETDS sets a given data set as the "current" data set.

USER INTERACTION:

The user is prompted for a treename and a node name. If neither exists within OLPARS, the user will be prompted for another value of each. If an initial colon is typed as the first character of the treename, the senior node of that tree will be used as the current node, and there will be no prompt for a node name.

EXAMPLE(S):

In the following examples, 'TESTTREE' is the current data tree name and '*****' is the current node name (denoting the senior node).

Session with SHORT prompts

TREE NAME? /TESTTREE/
NODE NAME? /*****/

(PROMPT NOTES: *****)

In the above example, if a colon ':' had been entered before the T in TESTTREE, the node name prompt would not have occurred. See the following example.

*****)

Session with LONG prompts

TYPE IN NAME OF OLPARS DATA TREE THAT IS TO
BE USED IN THE CURRENT DATA SET NAME
INITIAL ':' MEANS START AT SENIOR NODE
(8 CHARS. MAX.) - /:TESTTREE/

SETLOG

COMMAND NAME: SETLOG

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

SETLOG takes an existing logic tree and makes it the current logic tree (restores an old logic). The menu is then displayed.

USER INTERACTION:

The user is prompted for a treename. If the treename is an invalid name or the treename does not exist, the user will be asked for another treename.

EXAMPLE(S):

In the following examples, 'TESTTREE' is the name to be made current.

Session with SHORT prompts

TREENAME? /TESTTREE/

Session with LONG prompts

ENTER THE LOGIC TREENAME (MAXIMUM 8 CHARACTERS)?
/TESTTREE/

COMMAND NAME: SLCTMEAS

CATEGORY: MEASUREMENT EVALUATION COMMAND (subsidiary)

FUNCTIONAL DESCRIPTION:

SLCTMEAS allows the user to select the measurements to be used to create a new data tree during a data set transformation (see TRANSFRM). The current display must be a rank order display.

Measurements are selected either by entering a threshold, or by entering measurement numbers. If a threshold is entered, those measurements whose values are greater than the threshold are selected, if DSCRMEAS created the display, and those measurements whose values are less than the threshold are selected, if PROBCONF created the display. An asterisk precedes all measurements to be used in the data set transformation.

When individual measurements are specified, an asterisk will precede those measurements that have not been previously selected. That is, if a measurement has already been selected (via SLCTMEAS or UNION), and is chosen once again, it will no longer be selected for transformational use (the asterisk will no longer precede the measurement).

Note, SLCTMEAS can be used in excess measurement mode.

USER INTERACTION:

The user is asked if (s)he wants to select a threshold value or specific measurements. Depending upon which option is chosen, (s)he is prompted for those values.

EXAMPLE(S):

In the following example, the current rank order display is an overall ranking of measurements. Note, SLCTMEAS does not change the type of rank order display.

Session with SHORT prompts

(1) ENTER THRESHOLD
(2) ENTER MEASUREMENT(S)
SELECT AN OPTION (1 OR 2): /1/
THRESHOLD: /7.56/

SLCTMEAS (continued)

A RANK ORDER DISPLAY

DATE: 13-MAR-81 02:16:36

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| *4 | 8.3926E+00 | C | A/C |
| *2 | 8.3155E+00 | A | A/C |
| 1 | 7.5638E+00 | C | B/C |
| 3 | 6.9224E+00 | B | A/D |

(PROMPT NOTES: *****)

If there is no rank order display for the current data set, SLCTMEAS notifies the user and exits. If the user wishes to enter a threshold, the current rank order display must be either (1) overall, (2) measurement by class, or (3) measurement by class pair (see RANK). If the type of ranking is not 1, 2, or 3, SLCTMEAS notifies the user and exits. If the user wishes to enter specific measurements, the type of ranking is unimportant, but if the type of ranking is not 1, 2, or 3, the user will receive a message indicating that the current display will not be able to show which measurements have been selected. In this case, after SLCTMEAS is finished, the operator may then use RANK to see if measurements were correctly selected.

*****)

Session with LONG prompts

The following example assumes that the rank order display above is the current rank order display. Notice that measurement number four is not a selected measurement for transformation after SLCTMEAS is used this time.

OPTIONS:

(1) ENTER A THRESHOLD

(2) ENTER ONE OR MORE MEASUREMENT NUMBERS

TYPE A '1' OR A '2' DEPENDING ON THE DESIRED OPTION: /2/

ENTER THE MEASUREMENT NUMBER (END WITH 0) - /1/

ENTER THE MEASUREMENT NUMBER (END WITH 0) - /4/

ENTER THE MEASUREMENT NUMBER (END WITH 0) - /0/

SLCTMEAS (continued)

A RANK ORDER DISPLAY

DATE: 13-MAR-81 02:16:36

AN OVERALL RANKING

| MEAS | VALUE | CLASS | CLASS PAIR |
|------|------------|-------|------------|
| 4 | 8.3926E+00 | C | A/C |
| *2 | 8.3155E+00 | A | A/C |
| *1 | 7.5638E+00 | C | B/C |
| 3 | 6.9224E+00 | B | A/D |

SUMMCM

COMMAND NAME: SUMMCM

CATEGORY: UTILITY COMMAND

FUNCTIONAL DESCRIPTION:

SUMMCM displays a confusion matrix summary at the terminal.

USER INTERACTION:

NONE

COMMAND NAME: THRESHMOD

CATEGORY: LOGIC DESIGN COMMAND

FUNCTIONAL DESCRIPTION:

THRESHMOD modifies Fisher pairwise logic thresholds at a user chosen logic node. The user can change the number of thresholds for a particular class pair, and/or relocate the thresholds.

This command can only be used after FISHER has created pairwise logic in the current logic tree.

To evaluate the logic use PWEVAL.

USER INTERACTION:

The user is asked (1) which pairwise logic node should be modified, (2) which class pair should be modified, (3) which threshold to move (4) for the new position of the threshold.

EXAMPLE(S)

In the following example, the user has decided to modify pairwise logic node number 3, in particular, the class pair of 'soy' and 'oats'. Threshold 3 is moved to a new location. When satisfied, a comma is entered.

Session with SHORT prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3
NODE NUMBER? /3/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

THRESHMOD (continued)

ENTER 2 CLASS DISPLAY SYMBOLS
-OR- '*' FOR NEXT CLASS PAIR
-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /so/

(The projected vectors are displayed.)

THRESHOLD (1-5, 0 TO SAVE MODIFICATION
6 TO CHANGE NUMBER OF THRESHOLDS)? /3/

(Position the graphics cursor on the spot where the
threshold is desired.)

Type any alphanumeric key on the terminal (except 'Q')
to enter the threshold value into the current dataset.
A vertical line will be drawn on the display screen at
the threshold point.)

THRESHOLD (1-5, 0 TO SAVE MODIFICATION
6 TO CHANGE NUMBER OF THRESHOLDS)? /0/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS
-OR- '*' FOR NEXT CLASS PAIR
-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /,/

(The program puts up the menu)

(PROMPT NOTES)*****

When exiting from the class pair select prompt, a ','
is different from a <CR>. The ',' will save the changes
and cause the option to be changed to THRESHMOD. A <CR>
is interpreted as the standard OLPARS exit.

When entering a zero (0) to the threshold prompt, any
threshold modifications are saved. The next time this
particular class pair is viewed, the display scaling will
reflect the threshold modifications (i.e., histograms
may look different from the previous display).

When the graphics cursor is activated, a 'Q' response
(quit) is interpreted as the standard OLPARS exit.

In this example, the user modifies the next class pair of logic node 3, changes the number of thresholds for this class pair, and enters a comma when done.

Session with LONG prompts

LOGIC NODES WITH PAIRWISE LOGIC

2 3

ENTER THE LOGIC NODE YOU WISH TO MODIFY - /3/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS

-OR- '*' FOR NEXT CLASS PAIR

-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /*/

(The projected vectors are displayed.)

ENTER THE NUMBER OF THE THRESHOLD YOU WISH
TO MOVE (1-5, 0 TO SAVE MODIFICATION
6 TO CHANGE THE NUMBER OF THRESHOLDS) - /6/
NEW NUMBER OF THRESHOLDS - /4/

ENTER THE NUMBER OF THE THRESHOLD YOU WISH
TO MOVE (1-5, 0 TO SAVE MODIFICATION
6 TO CHANGE THE NUMBER OF THRESHOLDS) - /0/

(The screen is erased.)

CLASS SELECT LIST

soy oats weat corn

ENTER 2 CLASS DISPLAY SYMBOLS

-OR- '*' FOR NEXT CLASS PAIR

-OR- ',' TO EXIT WITH MODIFICATIONS SAVED

CLASS SYMBOLS: /,/

(The program puts up the menu)

TRANSFRM

COMMAND NAME: TRANSFRM

CATEGORY: MEASUREMENT EVALUATION COMMAND (subsidiary)

FUNCTIONAL DESCRIPTION:

TRANSFRM creates a new data tree whose structure is identical to the current data set using those measurements which have been "selected" during measurement evaluation. In a measurement evaluation (rank order) display, an asterisk precedes those measurements which have been "selected." Measurements are selected by running the commands 'SLCTMEAS' or 'UNION'.

Note, TRANSFRM can be used in excess measurement mode.

USER INTERACTION:

The user is prompted for the name of the OLPARS data tree to be created.

EXAMPLE(S):

Session with SHORT prompts

NEW TREE NAME? /PEACH/
TRANSFORMATION COMPLETE

(PROMPT NOTES: *****)

If the tree name entered by the user is the same as the current tree name, an error message is printed and 'TRANSFRM' remains in prompt mode. If the tree name entered already exists, but is not the same as the current tree name, the user is asked if s(he) wants to destroy the existing tree. If the user types 'Y' (for yes), 'TRANSFRM' creates a new tree with the user-specified name. If the user types 'N' (for no), 'TRANSFRM' asks for another tree name.

*****)

Session with LONG prompts

The following example assumes the name of the current data tree is 'APPLE'. The user also has a tree in her/his directory called 'MAPLE'. The user decides to destroy the tree 'MAPLE', so that the name 'MAPLE' can be used for the tree to be created as a result of the transformation.

TYPE IN NAME TO GIVE TO THE OLPARS DATA TREE
THAT WILL BE CREATED AS A RESULT OF THE TRANSFORMATION
(8 CHARS. MAX.) - /APPLE/
NEW TREE NAME MUST BE DIFFERENT
FROM CURRENT DATA SET TREE NAME
TYPE IN NAME TO GIVE TO THE OLPARS DATA TREE
THAT WILL BE CREATED AS A RESULT OF THE TRANSFORMATION
(8 CHARS. MAX.) - /MAPLE/
THE NAME "MAPLE"
IS IN USE IN YOUR DIRECTORY;
DO YOU WANT TO DESTROY THE DATA SET WITH THAT NAME (Y/N)?
/Y/
TRANSFORMATION COMPLETE

UNION

COMMAND NAME: UNION

CATEGORY: MEASUREMENT EVALUATION COMMAND (subsidiary)

FUNCTIONAL DESCRIPTION:

UNION performs a 'union by class' or 'union by class pair' function. The 'union by class' function uses the discriminant measurement vectors whose values separate one class from all other classes (class values). The 'union by class pair' function uses the discriminant measurement vectors whose values separate one class from another class (class pair values). For both functions, the discriminant values for each vector are ranked, and the measurement which corresponds to either the smallest or largest discriminant value has an asterisk placed next to it on the display. In the 'union by class' function, there are as many discriminant measurement vectors as there are classes. Thus, the maximum number of measurements that can have an asterisk placed next to them is the same as the number of classes in the data set. In the 'union by class pair' function, there are as many discriminant measurement vectors as there are class pairs. Therefore, the maximum number of measurements that can have an asterisk placed next to them is the same as the number of class pairs in the data set (see NOTE in 'RANK' User's Manual description defining 'class pairs'). In both cases, 'union by class' and 'union by class pair', the minimum number of measurements that can have an asterisk placed next to them is 1, because the same measurement could be the smallest (or largest) discriminant value in each vector. UNION is therefore the union (as in set notation) of the best measurements for separating one class from all other classes ('union by class') or for separating one class from another class ('union by class pair'). (NOTE: UNION does not erase selected measurements). All discriminant measurement values are ranked in ascending (PROBCONF) or descending (DSCRMEAS) order, depending on the 'CURRENT OPTION' (most recently executed Major Measurement Evaluation Command). UNION puts up an 'overall' ranking display.

Note, UNION can be used in excess measurement mode.

USER INTERACTION:

The user is asked to select one of the following options:
(1) union by class
(2) union by class pair

EXAMPLE(S):

Session with SHORT prompts

SELECT AN OPTION:
(1) UNION BY CLASS
(2) UNION BY CLASS PAIR
OPTION # = /1/

(PROMPT NOTES: *****)

UNION remains in prompt mode until the user types a
'1' or a '2' or the OLPARS character for exiting
commands.

*****)

Session with LONG prompts

TYPE A '1' OR A '2' DEPENDING ON THE DESIRED OPTION #:
(1) UNION BY CLASS
(2) UNION BY CLASS PAIR
OPTION NUMBER = /2/

References

- [1] Multics OLPARS Operating System, Final Technical Report, Volume I, RADC-TR-76-271, September, 1976.
- [2] D.H. Foley, "Considerations Of Sample And Feature Size", IEEE Transactions On Information Theory, Volume IT-18, No. 5, September, 1972, pp. 618-626.
- [3] Final Report of Project OLPARS V, Contract Number MDA 904-78-C-0576, January 5, 1980.
- [4] Hart, P.E., "The Condensed Nearest Neighbor Rule", IEEE Transactions On Information Theory, Volume IT-14, May 1968, pp. 515-576.
- [5] Sammon, J.W., "Interactive Pattern Analysis And Classification", IEEE Transactions On Computers, Volume C-19, Number 7, July 1970.

GLOSSARY

This glossary contains terms and phrases used throughout OLPARS. The letter in parenthesis, following the term or phrase, represents the subject material or context in which the term or phrase is used.

C - represents "command"

D - represents "display"

L - represents "logic tree"

T - represents "data tree"

Assigned Classes (L,D) - The classes in a logic tree at which vectors are "placed" as the result of a partial or an overall logic evaluation. A confusion matrix display shows how many vectors from the design (partial evaluation) or test (overall evaluation) data set were sent to each assigned class.

Associated Class Names - Data class names associated with a node in a logic tree. The associated class names indicate which data classes from the design data set are present at the logic node.

Between-Group Logic (L) - Usually referred to simply as "group logic." The logic defines regions in a multi-dimensional space, and each region may contain one or more classes. The logic does not necessarily break down a group of classes into single classes, as in the case of complete within-group logic, but instead distinguishes between groups of classes.

Class Display Symbol (D) - The first character of a class name. On a display, the character represents a data class or vector of a data class.

Class Pair (D) - Two data classes for which some measurement has been calculated (e.g. a Fisher direction or a discriminant measure). On a two-space display, the display flag symbol identifies class pairs. On a rank order display, the class pair is represented as "A/B", where "A" and "B" are the class display symbols of the two classes.

GLOSSARY

Class Select List - A list of class names or class display symbols presented to a user so that one or more may be selected for use by an OLPARS command.

Classification Table (L) - The table displayed at the terminal during classifier mode of overall logic evaluation. The table consists of three columns: (1) logic node, (2) associated classes, and (3) vector count (the number of vectors at a logic node). The table differs from a confusion matrix display in that the data classes of the vectors being evaluated are not shown, and no summary statistics are presented. A printed listing of the table may be obtained.

Cluster Plot (D) - A two-dimensional representation of N-space vectors, with each vector occupying a location within a grid.

Complete Within-Group Logic - Or called "within-group" logic mathematically defines a region for each class at a logic node. Therefore, within-group logic always completely defines logic for the group of classes on which it is designed.

Completed Logic Node (L) - A logic node which has only one class present.

Completed Logic Tree (L) - A logic tree in which all lowest nodes have only a single class present.

Confusion Matrix (D) - Summary, in table format, of the results of a partial or an overall logic evaluation.

Convex Point (D) - A user-selected point on the convex side of a boundary. The point is used during logic evaluation to determine on which side of a boundary a vector lies.

Convex Region (D) - A region, defined by a user drawn boundary, is convex if it lies completely on one side of any extended boundary segment.

Convex Side - The inside of the area delimited by the convex boundary.

Coordinate Projection (D) - Projection of each vector in a data set onto one (one-space) or two (two-space) vectors containing all zeros and a one in a user-selected position. The result for each vector is therefore one (one-space) or two (two-space) features which can be plotted.

Covariance Matrix (T) - A symmetric matrix which gives a measure of the variance between each pair of measurements in a data set.

Current Data Set (T) - The data set which has been most recently designated by the operator to be used by OLPARS commands.

- Current Logic (L)** - The logic tree which has been most recently designated by the operator to be used by OLPARS logic commands.
- Current Option (D)** - The name of the OLPARS command which most recently changed the operational or functional state of OLPARS. The current option is used to determine which menu (option list) should be displayed to the user.
- Data Class (T)** - A group of vectors representing a state in the environment.
- Data Partition (Boundary) (D)** - A convex boundary, consisting of a maximum of five line segments used in both structure analysis and logic design.
- Data Set Dimensionality (T)** - The number of features that comprise each vector of a data set.
- Data Vector (T)** - An object or an event in a state of the environment.
- Design Data Set (L)** - A data set for which logic has been designed.
- Discriminant Measure** - A measure of the discriminatory power of a set of features. This significance measure is particularly useful for ranking a set of features when the class conditional probability distributions are approximately unimodal. OLPARS uses three types of discriminant measures which are calculated by the command DSCRMEAS. The three types of discriminant measures are: (1) the discriminant measure for differentiating one class from another using a particular feature, (2) the discriminant measure for differentiating one class from all other classes using a particular feature, and (3) the discriminant measure for distinguishing all classes using a particular feature.
- Eigenvector Projection (T,D)** - A projection of all vectors in a data set onto one (one-space) or two (two-space) eigenvectors which correspond to user-selected eigenvalues.
- Excess Measurement Mode (T)** - A data set which has an excess number of measurements can only be operated on by a small subset of OLPARS commands. Those commands are said to run in excess measurement mode.
- Feature (Measurement) (T)** - A raw measurement or a statistic obtained from raw measurements.
- Feature Extraction (T)** - The process of defining features (measurements) that comprise the vector space.

GLOSSARY

Fisher Direction (Projection) (L) - An N-dimensional vector that is optimal for discriminating between two classes.

Global Scale (D) - In a one or two-space display, a scale that encompasses the entire data range.

Incomplete Logic (L) - A logic tree that contains incomplete logic nodes.

Incomplete Logic Node (L) - A lowest logic node for which there is more than one class present.

Logic - Mathematical algorithms which define regions to separate classes.

Logic Design (L) - The application of mathematical algorithms to a data set for the purpose of defining regions in a multi-dimensional space. Each region contains one or more of the data classes to which the algorithms have been applied.

Macro Plot (D) - A one-space display in which a frequency histogram for each data class is given on a separate baseline.

Mean Vector (T) - A vector composed of the means of the measurements (features) in a data set. Mean vectors are computed for each node of a data tree.

Measurement Evaluation (T) - Evaluation of the measurements in a data set based on the ability of the measurements to discriminate between data classes.

Measurement Transformation (T) - Creation of a new data tree using only those measurements which have been selected through the use of measurement evaluation routines (SLCTMEAS and UNION), or measurements explicitly created with the measurement transformation command (MEASXFRM).

Micro Plot (D) - A one-space display in which a frequency histogram for each data class is superimposed on a single base line.

Multimodal (T) - A data class is considered to be multimodal when vectors from the class cluster around more than one point (vector) in the class.

Nearest Mean Vector (L) - Logic which uses the class means as the main component for defining regions for the classes at a logic node. During evaluation, the distance between a vector and each class mean is calculated. The vector is assigned to the class which it is closest to in distance, unless the distance of the vector from the class exceeds a user-specified distance (in this case the vector is rejected).

- One-Space (D)** - Refers to a type of projection or an OLPARS display (plot) type. In a one-space projection, each vector in a data set is projected onto a user-selected projection vector. The result for each vector is a scalar (single value) which can be plotted along a single baseline (axis).
- Option List (Menu) (D)** - Although there is no restriction on the order in which OLPARS commands are invoked, commands should be run in a logical sequence (e.g. you would not attempt to evaluate logic before creating it). The OLPARS option list is displayed at the completion of each command to aid the user in selecting the next command to be invoked.
- Overall Logic Evaluation (L)** - Evaluation of an entire logic tree, beginning at the root (senior) node. Vectors from either the design data set, or from a test data set having the same dimensionality as the design data set, are tested at nodes along a path of the logic tree until all vectors are assigned to a lowest node.
- Overlap Graph (T)** - An overlap graph shows, for a particular measurement, the amount of overlap among the classes of a data set. The graph consists of stacked horizontal lines which are associated with data classes. Each line represents the range of the particular measurement for the data class being represented. The range of each class is scaled to the overall minimum and maximum values of the data set.
- Pairwise Logic (L)** - Logic which defines regions based on class pair statistics. Discriminating values are computed for all possible pairs of classes at a logic node. During evaluation, a vector is compared against the discriminating values for all class pairs. One class from each class pair receives a vote. The number of votes received by each class is tallied and the vector is assigned to the class with the most votes, providing a user-specified minimum vote count is satisfied. If the minimum vote count is not satisfied, the vector is rejected.
- Partial Logic Evaluation (L)** - Evaluation of vectors from the design data set at a single node in a logic tree. The vectors are assigned to nodes beneath the node at which the logic exists. A confusion matrix display is created.
- Projection** - Dot product of two vectors.
- Prompt Mode (C)** - Determines the type of requests for input (prompts) the user will receive from a command. The prompt mode may either be "short" or "long". In short prompt mode, the user receives short, terse prompts. In long prompt mode, the user receives prompts with more information about what should be typed. Users familiar with OLPARS will most likely prefer short prompts. The command CDEFAULT may be used to change the prompt mode.

GLOSSARY

Rank Order Display (D) - The type of display created by measurement evaluation commands.

Raw Measurement - Data taken directly from the environment. The measurement may or may not be a class discriminating feature.

Reassociated Class Name (L) - Class names associated with the lowest nodes of a completed logic tree. The reassociated class names may be different than the original design data class names. During an overall logic evaluation, using a test data set, the reassociated class names may be used to determine whether or not vectors are correctly classified. Therefore, the reassociated names are useful when the test data set class names are different from the design data set class names.

Rectangular Scale (D) - Refers to a two-space display in which the size of the units of the X and Y axes are different.

Restructure (T) - The process of subdividing a single data class into more classes.

Scale Mode (D) - Refers to the amount of the data range being displayed, either Global (entire range) or Zoom (sub-range).

Scale Type (D) - The type of scaling used on a one or two-space display. For a one-space display the scale type is either PROBABILITIES or COUNTS. For a two-space display the scale type is either SQUARE or RECTANGULAR.

Scatter Matrix (T) - A symmetric matrix which gives a measure of the variance between each pair of measurements in a data set. The scatter matrix for a data set differs from the covariance matrix in that the elements have not been divided by the total number of vectors in the data set. The elements of the matrix are therefore weighted by the number of vectors per class. It may be useful to use the scatter matrix of a data set when the results of calculations should reflect the fact that the data classes do not have the same number of vectors.

Scatter Plot (D) - A two-dimensional representation of N-space vectors, with each vector located at its natural projection point on the screen.

Senior Node (L,T) - The root node of a data or logic tree.

Square Scale (D) - Refers to a two-space display in which the value of the measurement units on both the X and Y axes are equal.

Structure Analysis - Projecting a data set onto a one or two-space plane for the purpose of determining if the structure of the data for a particular class is unimodal or multimodal. If one or more data classes are multimodal, partitions may be drawn to subdivide the classes and the data set can be restructured. The one and two-space projections may also be used as the basis for group logic design.

Test Data Set (L) - A user's data should be divided into two data sets. One data set should be used to design logic, and one to test the logic. The data set used to test the logic is called the test data set.

True Classes (D,L) - During logic evaluation, the classes in the data set being evaluated are called the true classes.

Two-Space (D) - Refers to a type of projection or an OLPARS display (plot) type. In a two-space projection, each vector in a data set is projected onto two user-selected projection vectors. The result of the first projection is a scalar which becomes an X coordinate. The result of the second projection is a scalar which becomes a coordinate. Each X, Y point can be plotted along the X and Y axes.

Unimodal - A data class is considered to be unimodal when vectors from the class cluster around one point (vector) in the class.

Union By Class - Refers to a measurement evaluation function which ranks measurements for each class and selects, from each ranking, the best measurement. The result is the union of the best measurements for distinguishing single classes from all other classes.

Union By Class Pair - Refers to a measurement evaluation function which ranks measurements for each class pair and selects from each ranking, the best measurements. The result is the union of the best measurements which distinguish one class from another.

Vector Identifier (T) - A numeric value assigned to a vector for the purpose of identifying it.

Zoom Scale (D) - In a one or a two space display, a scale that represents a subsection of the original display.

APPENDIX A
MULTICS OLPARS MATHEMATICS

This appendix contains a copy of the mathematical section found in reference [1].

1.3 MOOS MATHEMATICS

This section of the report presents the mathematical justification or explanation of the algorithms used in MOOS. Only the general explanation is included here; the reader is referred to Section 3 for details concerning algorithm implementation.

1.3.1 Measurement Evaluation

In solving a pattern classification problem, the researcher will often be concerned with the discriminatory qualities of the L measurements. In general, it is desirable to use the minimum number of measurements that achieves a satisfactory solution. To this end, the MOOS system provides three (3) methods for ranking the discriminatory power of a set of L measurements.

If desired, the rankings may be used as the basis for a measurement reduction transformation to a subset consisting of the M most discriminatory measurements. An optimal method for selecting a subset of M measurements must involve a consideration of the decision logic criterion, such as the Bayes Risk or the probability of error. This, in turn, requires the estimation of the joint probability functions for all possible n-tuples. The obvious computational difficulties in obtaining an optimal ranking preclude this approach in all but the simplest problems. Therefore, the following sub-optimal algorithms are provided as options to rank-order the L measurements x_1, x_2, \dots, x_L . Each algorithm provides three distinct types of rankings. The first uses a significance measure of a particular component, e.g. x_p , for discriminating class i from class j; this significance will be designated by $M_{ij}(x_p)$. The second type of ranking uses a significance measure of x_p for discriminating class i from all other classes, and is designated $M_i(x_p)$. The last type of ranking uses a measure of the overall significance of x_p for discriminating all classes, and is designated $M(x_p)$.

1.3.1.1 The Discriminant Measure

This algorithm is implemented in the MOOS function dscrmeas. This significance measure is particularly useful for ranking the L measurements when the class conditional probability distributions are approximately unimodal. The discriminant measure for differentiating class i from class j using measurement x_p is defined as:

$$M_{ij}(x_p) = \frac{\left[\bar{x}_p^{(i)} - \bar{x}_p^{(j)} \right]^2}{\left[(N_i - 1) \left[\hat{\sigma}_p^{(i)} \right]^2 + (N_j - 1) \left[\hat{\sigma}_p^{(j)} \right]^2 \right]}$$

where

$\bar{x}_p^{(j)}$ = the estimated mean of class j along measurement x_p

$\hat{\sigma}_p^{(j)}$ = the estimated standard deviation of class j along measurement x_p

N_i = the number of vectors in class i

The discriminant measure for differentiating class i from all other classes using measurement x_p is defined as:

$$M_i(x_p) = \sum_{j \neq i}^K M_{ij}(x_p)$$

Finally, the discriminant measure for distinguishing all classes using measurement x_p is defined as:

$$M(x_p) = \sum_{i=1}^K M_i(x_p) = \sum_{i=1}^K \sum_{j \neq i}^K M_{ij}(x_p)$$

1.3.1.2 The Probability of Confusion Measure

This algorithm is implemented in the MOOS function probconf.

This measure is recommended when the assumption of class unimodality cannot be justified. It is valid for any probability distribution since it essentially measures the overlap of the class conditional probabilities. Computationally, it is much more complex than the previous measure.

Let x_p designate the measurement under evaluation and $P(x_p/C_j)$, $j = 1, 2, \dots, k$ be the marginal class conditional probability distributions. Next, consider the distributions for the two classes i and j shown in Figure 1 - 14. The measure for differentiating class i from class j using x_p is defined as follows:

$$M_{ij}(x_p) = \int_{-\infty}^{\theta_1} P(x_p/C_j) dx_p + \int_{\theta_1}^{\theta_2} P(x_p/C_i) dx_p + \int_{\theta_2}^{+\infty} P(x_p/C_j) dx_p$$

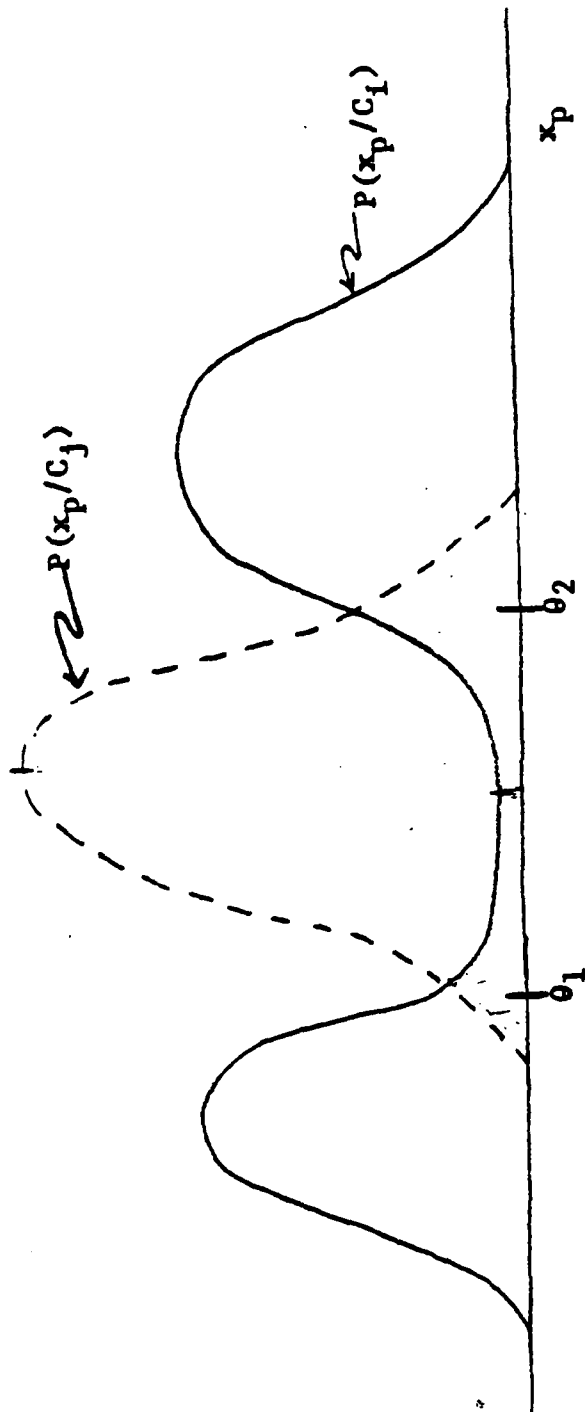


FIGURE 1-14

Marginal Class Conditional Probability
Distributions (2 Classes)

Since the functional forms of the class conditional probabilities are not known, we estimate the marginal class distributions using the sample data. This method makes use of histogram approximations like those shown in Figure 1-15. A detailed discussion of histogram computation will be presented later.

The measurement x_p will be divided into cells of width Δ . The probability that a sample from class j will occupy the r th cell along measurement x_p is given by

$$P_{rp}^{(j)} = \int_{\text{rth cell}} P(x_p/C_j) dx_p$$

Thus, the pairwise measure for differentiating class i from class j can be computed by:

$$M_{ij}(x_p) = \sum_{r=1}^{N_p} \min_{i,j} \left\{ P_{rp}^{(i)}, P_{rp}^{(j)} \right\}$$

The measure for differentiating class i from all other classes using x_p is defined by:

$$M_i(x_p) = \sum_{j \neq i}^K M_{ij}(x_p)$$

Finally, the overall measure of significance of x_p for differentiating all classes is computed as follows:

$$M(x_p) = \sum_{i=1}^K M_i(x_p) = \sum_{i=1}^K \sum_{j \neq i}^K M_{ij}(x_p)$$

The discriminant measure is the simplest measure and therefore is the fastest to compute. However, it can produce misleading results when the data classes are not unimodal. Consider, for example, the two marginal distributions shown in Figure 1-16. The discriminant measure for X is quite small, since the separation between the class means relative to the sum of their variances is small; however, measurement X yields excellent between-class discrimination. This weakness is not a problem with the probability of confusion algorithm, since this latter is relatively independent of the functional form of the class distributions.

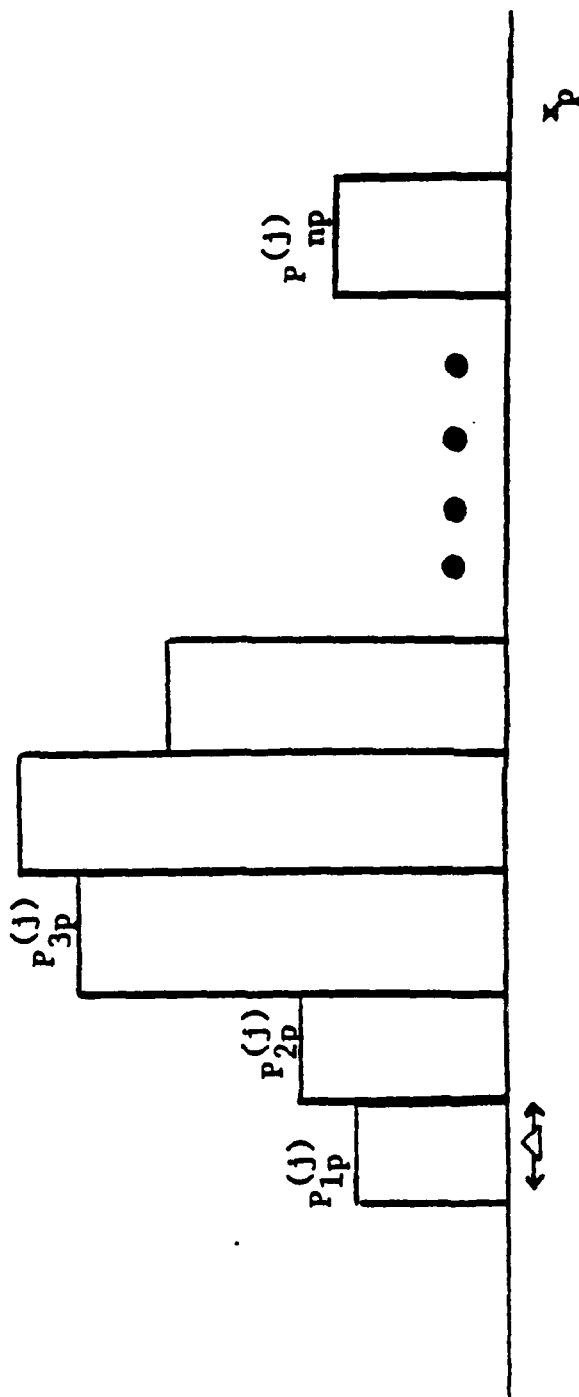


FIGURE 1-15

Histogram Approximations of Marginal Class Distributions

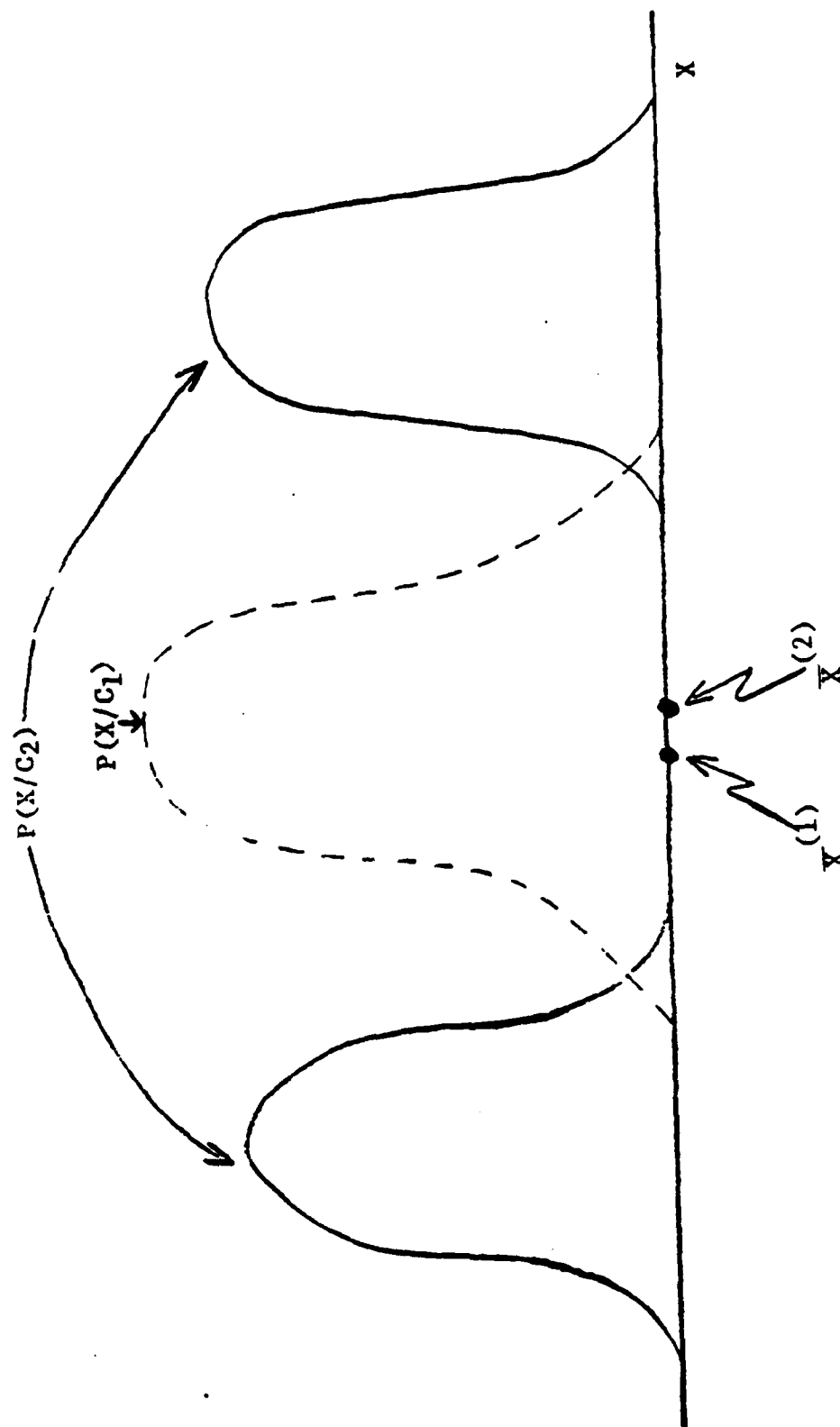


Figure 1-16 Example of Marginal Class Probability Distributions for which the Discriminant Measurement Algorithm Produces Misleading Results

1.3.1.3 Higher-Order Measurement Evaluation

An algorithm which evaluates higher-order combinations of measurements is implemented in the MOOS function features. A problem inherent with both the discriminant measure and the probability of confusion computations is that each measurement is treated alone. The features algorithm may evaluate one or more measurements at a time using the divergence measure as its criterion.

The divergence measure is useful in subspace feature evaluation when the underlying distributions are multivariate normal or when the underlying distributions are unimodal (4,5). An advantage of the divergence measure over a discriminant measure or probability of confusion significance measure is that it considers the correlation between features.

The divergence J is defined as:

$$J = \int [p(X/\omega_i) - p(X/\omega_j)] \log \left[\frac{p(X/\omega_i)}{p(X/\omega_j)} \right] dx$$

where $p(X/\omega_i)$ is the class conditional probability distribution for any set of measurements X .

Let $p(X/\omega_i)$ be Gaussian, i.e.,

$$p(X/\omega_i) \sim N(\mu_i', \Sigma_i) \quad i = 1, 2$$

where μ_i' are the means and Σ_i are the covariance matrices of the patterns in classes ω_i . For Gaussian-distributed pattern classes, this becomes:

$$J = 1/2(\mu_i' - \mu_j')^T [\Sigma_i^{-1} + \Sigma_j^{-1}] (\mu_i' - \mu_j') + 1/2 \text{ trace}(\Sigma_i^{-1}\Sigma_j + \Sigma_j^{-1}\Sigma_i - 2I)$$

The divergence measure for differentiating class i from class j using measurements x_p, \dots, x_q is obtained by evaluating the above expression for the subspace defined by measurements x_p, \dots, x_q .

$$M_{ij}(x_p, \dots, x_q) = J(x_p, \dots, x_q)$$

The measure for differentiating class i from all other classes using measurements x_p, \dots, x_q is defined as:

$$M_i(x_p, \dots, x_q) = \sum_{j \neq i}^K M_{ij}(x_p, \dots, x_q)$$

The measure for distinguishing all classes using measurements x_p, \dots, x_q is defined as:

$$M(x_p, \dots, x_q) = \sum_{i=1}^K M_i(x_p, \dots, x_q) =$$

$$\sum_{i=1}^K \sum_{j \neq i}^K M_{ij}(x_p, \dots, x_q)$$

Since it is not practical to evaluate all possible combinations of the L measurements to determine an optimal feature subspace, a number of suboptimal search procedures are available.

The forward sequential suboptimal search procedure finds the "best" subset of N from the original L measurements using the measure for distinguishing all classes, $M(x_p, \dots, x_q)$. The first measurement selected is the best of the L measurements taken one at a time. The second measurement selected is the best of the $L-1$ remaining measurements when taken in combination with the first selected measurement. The third measurement selected is the best of the $L-2$ remaining measurements when taken in combination with the first and second selected measurement, and so on. The procedure halts when the user-specified value N is reached.

The union best by class approach to measurement selection utilizes the measure for distinguishing class i from all other classes, $M_i(x_p, \dots, x_q)$. The procedure is quite similar to the forward sequential technique except that at each step, more than one measurement may be selected. On the first round, the best measurement for distinguishing each of the K classes is selected, i.e., anywhere from 1 to K different measurements may be selected. The second round takes all remaining measurements in combination with the previously selected measurements, and again may add anywhere from 1 to K new measurements. The procedure halts when the user-specified value N is reached or exceeded.

The union best by class pair approach to measurement selection utilizes the measure for distinguishing class i from class j , $M_{ij}(x_p, \dots, x_q)$. The search algorithm is almost identical to the union best by class procedure. On the first round the best measurement for distinguishing each of the possible class pairs is selected, i.e., anywhere from 1 to $K(K-1)/2$ measurements may be selected. The second round (and all subsequent rounds) takes all the remaining measurements in combination with the previously selected measurements, and again

may add anywhere from 1 to $K(K-1)/2$ new measurements. The procedure halts when the user-specified value N is reached or exceeded.

The measurement selection procedure has a great deal of flexibility in that the previously described techniques may be interactively mixed in any sequence. Furthermore, a preferred subset of the feature space may be selected as a starting point for the measurement selection computations.

1.3.2 Structure Analysis

The basic use of structure analysis in MOOS is in determining if the structure of the data for a particular class is unimodal or multimodal. If it is multimodal, it is frequently better to subdivide the class before attempting to design logic for distinguishing between classes. This is particularly true if the logic to be designed is statistically based.

All of the algorithms for structure analysis in MOOS involve projecting the data onto a one-or two-space and allowing the analyst to draw a partition(s) of the space if multimodality is present. All of the projections except one, NLM, are linear. The linear projections may also be used as the basis for group logic design.

No justification or explanation is given for coordinate or arbitrary vector projections.

1.3.2.1 The Eigenvector Plane (Least Squares)

The following section is an explanation and proof of the contention that planes defined by the two eigenvectors corresponding to the two largest eigenvalues of the estimated covariance matrix are optimal by the least squares criterion.

This can be shown as follows:

1. Define a plane by two unit orthogonal vectors \underline{e}_1 and \underline{e}_2 through a shifted origin denoted by \underline{d} (\underline{d} will turn out to be the mean of the data).
2. Set up an expression for the error (E) that arises in fitting the data by the plane described in (1). This will be obtained by summing the squared residuals from the plane.
3. Next, minimize the error E with respect to \underline{d} , \underline{e}_1 and \underline{e}_2 , under the constraints that \underline{e}_1 and \underline{e}_2 be unit vectors and orthogonal.
4. \underline{d} will be found to be the mean vector. The eigenvectors of the estimated covariance matrix will be shown to be solutions to the minimization problem and, particularly, the two eigenvectors corresponding to the largest eigenvalues will turn out to be the desired solution. Note that there exists an infinite number of solutions, since any orthogonal rotation of \underline{e}_1 , \underline{e}_2 in the plane defined by \underline{e}_1 and \underline{e}_2 will also be a solution; however, all these solutions describe the same least squares plane.

Let $\left. \begin{matrix} \underline{X}_1 \\ \vdots \\ \underline{X}_N \end{matrix} \right\}$ be L-dimensional data vectors

\underline{d} = new origin for the data

$\underline{e}_1, \underline{e}_2$ define the plane through the new origin.

Define $\underline{Y}_1 = \underline{X}_1 - \underline{d}$

The residual distance squared from the fitting plane for the kth data vector is given by

$$\left| \underline{r}_k \right|^2 = \underline{r}_k^t \underline{r}_k = \left| \underline{Y}_k - (\underline{Y}_k^t \underline{e}_1) \underline{e}_1 - (\underline{Y}_k^t \underline{e}_2) \underline{e}_2 \right|^2$$

The fitting error is given by the summation of the squared residual, i.e.

$$E = \sum_{k=1}^N \underline{r}_k^t \underline{r}_k = \sum_{k=1}^N \left\{ \underline{Y}_k^t \underline{Y}_k - (\underline{Y}_k^t \underline{e}_1)^2 - (\underline{Y}_k^t \underline{e}_2)^2 \right\}$$

Using Lagrange multipliers to account for the constraints on \underline{e}_1 and \underline{e}_2 we obtain

$$E^* = \sum_{k=1}^N \left\{ \underline{X}_k^t \underline{X}_k - 2 \underline{X}_k^t \underline{d} + \underline{d}^t \underline{d} - (\underline{X}_k^t \underline{e}_1 - \underline{d}^t \underline{e}_1)^2 - (\underline{X}_k^t \underline{e}_2 - \underline{d}^t \underline{e}_2)^2 \right\} \\ - \lambda_1 \left[\underline{e}_1^t \underline{e}_1 - 1 \right] - \lambda_2 \left[\underline{e}_2^t \underline{e}_2 - 1 \right] - \lambda_3 \underline{e}_1^t \underline{e}_2$$

Taking the partial with respect to \underline{d} we obtain

$$0 = \frac{\partial E^*}{\partial \underline{d}} = \sum_{k=1}^N \left\{ -2 \underline{X}_k + 2 \underline{d} + 2 (\underline{X}_k^t \underline{e}_1 - \underline{e}_1^t \underline{d}) \underline{e}_1 + 2 (\underline{X}_k^t \underline{e}_2 - \underline{e}_2^t \underline{d}) \underline{e}_2 \right\}$$

$$\text{Let } \underline{U} = \frac{1}{N} \sum_{k=1}^N \underline{X}_k$$

Substitute

$$\frac{\partial E^*}{\partial \underline{d}} = 0 = 2N \left[(\underline{U}-\underline{d}) + \left\{ (\underline{U}-\underline{d})^t \underline{e}_1 \right\} \underline{e}_1 + \left\{ (\underline{U}-\underline{d})^t \underline{e}_2 \right\} \underline{e}_2 \right]$$

$\therefore \underline{U}=\underline{d}$ is a solution. Thus \underline{d} is the data mean.

continuing

$$\frac{\partial E^*}{\partial \underline{e}_1} = \left\{ 2 \sum_{k=1}^N \left[\underline{y}_k \underline{y}_k^t \right] \underline{e}_1 \right\} - 2\lambda_1 \underline{e}_1 - \lambda_3 \underline{e}_2 = 0$$

$$\frac{\partial E^*}{\partial \underline{e}_2} = \left\{ 2 \sum_{k=1}^N \left[\underline{y}_k \underline{y}_k^t \right] \underline{e}_2 \right\} - 2\lambda_2 \underline{e}_1 - \lambda_3 \underline{e}_1 = 0$$

Note that the estimated covariance matrix $\underline{\Sigma}$ is given by

$$\underline{\Sigma} = \frac{1}{n-1} \sum_{k=1}^N \left[\underline{y}_k \underline{y}_k^t \right]$$

Let $\hat{\underline{\Sigma}} = (N-1) \underline{\Sigma}$, and substitute above

$$2 \hat{\underline{\Sigma}} \underline{e}_1 - 2\lambda_1 \underline{e}_1 - \lambda_3 \underline{e}_2 = 0 \quad (A)$$

$$2 \hat{\underline{\Sigma}} \underline{e}_2 - 2\lambda_2 \underline{e}_2 - \lambda_3 \underline{e}_1 = 0 \quad (B)$$

Multiply (A) from the left by \underline{e}_1^t

Multiply (B) from the left by \underline{e}_2^t

Multiply (A) from the left by \underline{e}_2^t to obtain

$$\underline{e}_1^t \hat{\underline{\Sigma}} \underline{e}_1 = \lambda_1 \quad (1)$$

$$\underline{e}_2^t \hat{\underline{\Sigma}} \underline{e}_2 = \lambda_2 \quad (2)$$

$$2 \underline{e}_2^t \hat{\underline{\Sigma}} \underline{e}_1 = \lambda_3 \quad (3)$$

Substitute back into (A) and (B)

$$(A) \quad 2 \sum \hat{e}_1 - 2 \left[\underline{e}_1^t \sum \hat{e}_1 \right] \underline{e}_1 - 2 \left[\underline{e}_2^t \sum \hat{e}_1 \right] \underline{e}_2 = 0$$

$$(B) \quad 2 \sum \hat{e}_2 - 2 \left[\underline{e}_2^t \sum \hat{e}_2 \right] \underline{e}_2 - 2 \left[\underline{e}_2^t \sum \hat{e}_1 \right] \underline{e}_1 = 0$$

Now let $\sum \hat{e}_1 = \alpha_1 \underline{e}_1$ and

$\sum \hat{e}_2 = \alpha_2 \underline{e}_2$ and substitute

into (A) and (B)

$$(A) \Rightarrow 2 \alpha_1 \underline{e}_1 - 2 \alpha_1 \underline{e}_1 = 0$$

$$(B) \Rightarrow 2 \alpha_2 \underline{e}_2 - 2 \alpha_2 \underline{e}_2 = 0$$

• • A solution plane is given by the two eigenvectors of the estimated covariance matrix \sum .

The least squared error is given by

$$E = R - \underline{e}_1^t \sum \hat{e}_1 - \underline{e}_2^t \sum \hat{e}_2$$

$$\text{where } R = \text{constant} = \sum_{k=1}^n \underline{y}_k^t \underline{y}_k$$

or equivalently

$$E = R - \alpha_1 - \alpha_2$$

Therefore, the error is minimized by selecting the two eigenvectors corresponding to the two largest eigenvalues.

It can similarly be shown that the projection on the eigenvector associated with the largest eigenvalue is the best (by the least squares criterion) one-space projection.

1.3.2.2 Discriminant Projections

The MOOS functions ardg\$ and asdg\$ offer the analyst another projection direction or plane. The only difference between the two functions is in how the two classes upon which the projection is based are determined. The two classes may be composed of any two classes of the current data set or they may be composed of any two groups of classes which are "lumped together" for the purpose of determining the projection direction(s).

The entire current data set is projected into the space defined by the Fisher Discriminant \underline{d}_1 and a second vector \underline{d}_2 , where \underline{d}_2 is that direction which maximizes the projected between-class scatter relative to the sum of the projected within-class scatter, under the constraint that \underline{d}_2 be orthogonal to \underline{d}_1 . In summary,

$$\underline{d}_1 = \alpha_1 W^{-1} \underline{\Delta}$$

$$\underline{d}_2 = \alpha_2 \left[W^{-1} - (\underline{\Delta}^T [W^{-1}]^2 \underline{\Delta} / \underline{\Delta}^T [W^{-1}]^3 \underline{\Delta}) [W^{-1}]^2 \right] \underline{\Delta}$$

where α_1 and α_2 are normalizing constants

$\underline{\Delta}$ = the difference between the class mean vectors, $\mu_1 - \mu_2$

W = sum of the within-class scatter matrices

Notice that both \underline{d}_1 and \underline{d}_2 are computed using W^{-1} . If the data lies in a subspace, then it can be shown that W will be singular. If the data is approximately contained in any subspace, then W will at best be ill-conditioned. In either case, the numerical computation of W^{-1} will be extremely tenuous. Thus, prior to computing W^{-1} , W must first be checked to determine if it is ill-conditioned or singular. In either case, we will compute a subspace such that when the data is orthogonally projected onto this subspace, the $W_{\text{new}} = TW_{\text{old}}T^T$ will be well-conditioned. Next, \underline{d}_1 and \underline{d}_2 will be computed in the subspace using W_{new} , and finally, we will transform \underline{d}_1 and \underline{d}_2 back to the original L-dimensional space.

If the one-space option is chosen the data is projected on \underline{d}_1 , that is, in the Fisher direction only.

1.3.2.3 Generalized Discriminant Projections

The MOOS function gndv\$ offers the analyst the capability of projecting data onto a discriminant direction or plane which has been optimized to produce maximum discrimination for all classes. This is a generalization of the Fisher discriminant projection described in Section 1.3.2.2.

The Fisher discriminant is obtained by solving for the unit vector d which maximizes the following ratio:

$$R = \frac{d^T B d}{d^T W d}$$

where B is the between-class scatter matrix, and W is the sum of the within-class scatter matrices.

To solve for the generalized Fisher discriminant directions, we take the vector derivative of the above ratio R with respect to d and set the resultant equation to zero. The procedure generates the following generalized eigenvector equation.

$$\begin{aligned} [B - \lambda W] d &= 0 \\ [W^{-1}B - \lambda I] d &= 0 \end{aligned}$$

The generalized discriminant vectors are the eigenvectors of the non-symmetric matrix $W^{-1}B$. The rank of the between-class scatter matrix for the K -class discrimination problem is $K-1$, therefore, no more than $K-1$ nonzero eigenvector solutions exist. Thus, the generalized discriminant vector function produces $K-1$ discriminant vectors, with the vectors which correspond to the largest eigenvalues producing the maximum discrimination. The Gram-Schmidt orthonormalization technique is applied to the eigenvectors to insure that they are orthogonal unit vectors (6).

1.3.2.4 Nonlinear Mapping

The Nonlinear Mapping Algorithm (NLM) is based upon a point mapping of the N L -dimensional vectors from the L -space to a lower-dimensional space such that the inherent structure of the data is approximately preserved under the mapping. The approximate structure preservation is accomplished by fitting N points in the lower-dimensional space such that their interpoint distances approximate the corresponding interpoint distances in the L -space.

Suppose that we have N vectors in an L -space designated X_i , $i = 1, \dots, N$ and, corresponding to these, we define N vectors in the two-space designated Y_i , $i = 1, \dots, N$. Let the distance between the vectors X_i and X_j in the L -space be defined by

$$d_{ij}^* = \text{dist } [X_i, X_j]$$

and the distance between the corresponding vectors Y_i and Y_j in the two-space be defined by

$$d_{ij} = \text{dist } [Y_i, Y_j]$$

Let us now randomly¹ choose an initial two-space configuration for the Y vectors and denote this configuration as follows:

$$Y_1 = \begin{bmatrix} y_{11} \\ y_{12} \end{bmatrix} \quad Y_2 = \begin{bmatrix} y_{21} \\ y_{22} \end{bmatrix} \quad \dots \quad Y_N = \begin{bmatrix} y_{N1} \\ y_{N2} \end{bmatrix}$$

¹ - For the purpose of this discussion it is convenient to think of the starting configuration as being selected randomly; however, in practice the initial configuration for the vectors is found by projecting the L -dimensional data orthogonally onto a two-space spanned by the two original coordinates with largest variances.

Next we compute all the two-space interpoint distances d_{ij} , which are then used to define an error E ; E is a measure of how well the present configuration of N points in the two-space fits the N points in the L -space, i.e.,

$$E = \frac{1}{\sum_{i < j} [d_{ij}^*]} \sum_{i < j}^N \frac{[d_{ij}^* - d_{ij}]^2}{d_{ij}^*} .$$

Note that the error is a function of the $2 \cdot N$ variables y_{pq} , $p = 1, \dots, N$ and $q = 1, 2$. The next step in the NLM algorithm is to adjust the y_{pq} variables or, equivalently, change the 2-space configuration so as to decrease the error. We use a steepest descent procedure to search for a minimum error.

Let $E(m)$ be defined as the mapping error after the m th iteration, i.e.,

$$E(m) = \frac{1}{c} \sum_{i < j}^N [d_{ij}^* - d_{ij}(m)]^2 / d_{ij}^*$$

where

$$c = \sum_{i < j}^N [d_{ij}^*]$$

and

$$d_{ij}(m) = \sqrt{\sum_{k=1}^2 [y_{ik}(m) - y_{jk}(m)]^2} .$$

The new 2-space configuration at time $m + 1$ is given by

$$y_{pq}(m + 1) = y_{pq}(m) - (MF) \cdot \Delta_{pq}(m)$$

where

$$\Delta_{pq}(m) = \frac{\partial E(m)}{\partial y_{pq}(m)} \left/ \frac{\partial^2 E(m)}{\partial y_{pq}(m)^2} \right|$$

and MF is the "magic factor" which was determined empirically to be $MF \approx 0.3$. The partial derivatives are given by

$$\frac{\partial E}{\partial y_{pq}} = -\frac{2}{c} \sum_{\substack{j=1 \\ j \neq p}}^N \left[\frac{d_{pj}^* - d_{pj}}{d_{pj} d_{pj}^*} \right] (y_{pq} - y_{jq})$$

and

$$\frac{\partial^2 E}{\partial y_{pq}^2} = -\frac{2}{c} \sum_{\substack{j=1 \\ j \neq p}}^N \frac{1}{d_{pj}^* d_{pj}} \left[(d_{pj}^* - d_{pj}) - \frac{(y_{pq} - y_{jq})^2}{d_{pj}} \left(1 + \frac{d_{pj}^* - d_{pj}}{d_{pj}} \right) \right]$$

In our program we take precautions to prevent any two points in the two-space from becoming identical. This prevents the partials from "blowing up."

Because the number of computations required in the NLM algorithm is approximately proportional to $N^2/2$ (where N is the number of vectors), the MOOS implementation of NLM has an upper limit of 200 vectors. If the number of vectors in the current data set exceeds 200, a reduction is required; the algorithm for performing this reduction is explained below.

The user specifies the number of vectors (or cluster centers) to which he wishes the set to be reduced. If the current data set contains more than one class, the user also specifies whether he wants the number of vectors in each class of the new set to be the same, or proportional to the number of vectors in each class of the original data set. The reduction process is then performed on one class at a time.

Given that M is the number of vectors in the original class, and N ($N < M$) is the number of vectors desired for the reduced class, then the number of original vectors lumped together to produce a reduced vector is $M/N=K$. Each of the N reduced vectors is thus the mean vector of K vectors from the original class. The selection as to which K vectors are to be clustered together is made as follows: The entire set of vectors is searched; the vector which is farthest away (in the Euclidean sense) from the mean of the entire class is picked as a starting vector. Then the $K-1$ closest vectors to that starting vector are found, and the mean vector of those $1+(K-1)=K$ vectors is taken as the reduced vector or cluster center. The starting vector for the next cluster center is found from among the remaining vectors by searching for the one furthest away from the previous cluster; the clustering process is repeated as often as necessary. (If K is not an integer, then the first A ($A < N$ where A is the remainder of M/N) cluster centers will be the mean of $[K] + 1$ vectors and the remaining $N-A$ cluster centers will be the mean of $[K]$ vectors.)

1.3.3 Logic Design

In general, the primary goal of a pattern classification analyst is to design a logic, or series of tests, which will, with a suitable degree of accuracy, assign an unlabeled vector from the feature space to a particular class (or reject it as unclassifiable within the required degree of probability).

MOOS provides the analyst with several types of logic design algorithms and variations within each type.

1.33.1 Group Logic Design

In group logic, the analyst makes an interactive, subjective decision and actually participates in the logic design process. The particular node of the logic tree for which logic is being designed is examined; the vectors from the classes present there are projected on a one- or two-space. If there is (in the analyst's judgment) sufficient separation between classes, or between groups of classes, he may draw one or two boundaries so that the feature space is partitioned into two or three regions. These regions are then labeled as to the class or classes present in them (a region may be labeled as the null class or reject region). This is illustrated in Figure 1-17.

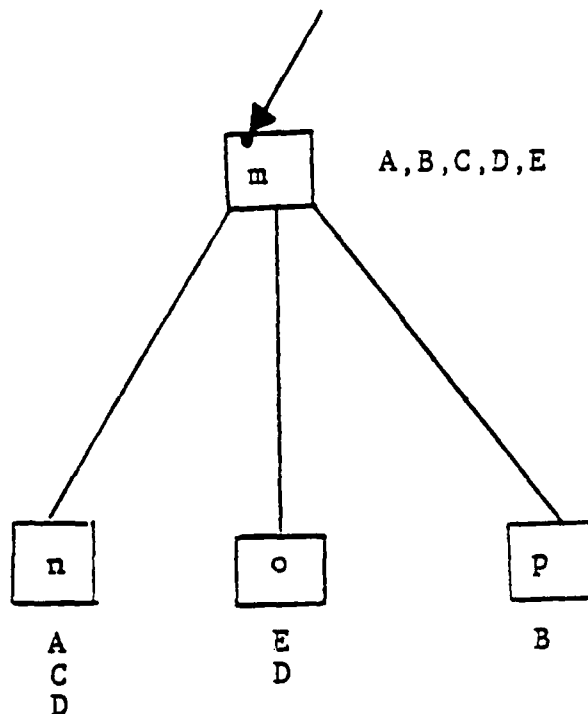


FIGURE 1-17 LOGIC TREE NODE - GROUP LOGIC

In this example, group logic which was designed at node m separated the five classes present (A,B,C,D, and E) into three groups. Class B was completely separable from the other classes and was assigned to node p of the logic tree; the remaining portions of the feature space, assigned to nodes n and o, contain the groups of classes A,C and D, and E and D respectively. Notice that the samples from class D fell into both regions; this is permissible.

For the one-space implementation of these logics, the mathematics is extremely simple. The unlabeled vector to be classified is merely projected (dot product) onto the projection direction (discriminant); the value of this scalar is then compared to the value of the boundary (threshold) drawn by the user.

Two-space logic mathematics is slightly more complicated; it is illustrated below.

When the user defines a two-space boundary, he draws, on the projection plane, from one to five connected line segments which must define a convex region; he then draws a reference point indicating which is the convex side. See Figure 1-18 on the following page.

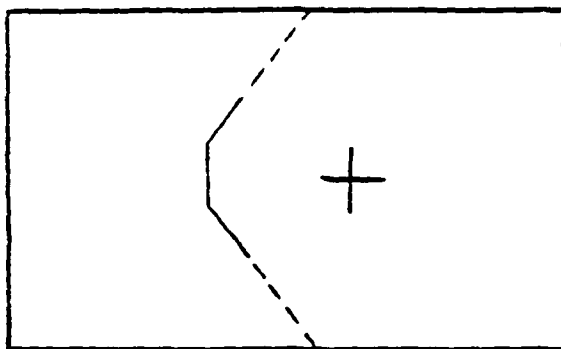


FIGURE 1-18

3-Line Segment, 2-Space Boundary.

The transition from the boundary drawn on the two-space projection to the mathematical logic creates a sequence of discriminant vectors and thresholds, one pair for each line segment. In the evaluation of the logic, the unlabeled vector is projected on each discriminant in turn and is then compared to the threshold. If it is less than the threshold for any given line segment, the vector is on the non-convex side of the partition; if it is greater than the threshold, it is on the convex side of the partition. The determination of the discriminant and of the threshold for a line segment follows.

Given three points on the projected plane

point S (boundary start point)
 point E (boundary end point)
 point C (point on "convex" side of boundary)

if these are considered as vectors (in the projected two-space), i.e.

$$\underline{S} = \langle x_S, y_S \rangle$$

$$\underline{E} = \langle x_E, y_E \rangle$$

$$\underline{C} = \langle x_C, y_C \rangle$$

then a vector \underline{D} normal to the boundary line in the projected two-space is given by: $\langle d_1, d_2 \rangle$ where $d_1 = + (y_E - y_S)$ and $d_2 = -(x_E - x_S)$.

Project the convex point and boundary end point onto this vector.

Let $P_C = \underline{C} \cdot \underline{D}$

$P_E = \underline{E} \cdot \underline{D}$

Then if $P_C \geq P_E$ save P_E as the discriminant threshold and compute and store the discriminant vector.

or if $P_C < P_E$ replace \underline{D} by $\langle -d_1, -d_2 \rangle$, save $-P_E$ as the discriminant threshold and compute the discriminant vector.

The discriminant vector = $d_1 \underline{X} + d_2 \underline{Y}$, where \underline{X} and \underline{Y} are the L-dimensional projection vectors used to project a point in L-space onto the projection plane (on which the boundary was drawn) for the purpose of evaluating logic (i.e. deciding if any point V in L-space lies on the "convex" side of the L-space hyperplane determined by the boundary $(\underline{E}-\underline{S})$ drawn on the projection plane).

If $\underline{V} \cdot \underline{A} \geq \text{threshold}$

where \underline{V} = L-dimensional vector for point V , and

\underline{A} = the discriminant vector,

then the point V is on the convex side of the boundary.

1.3.3.2 Complete Within-Group Logic

This type of logic creates a node, or region within the feature space, for each individual class present at the node for which logic is being designed. The logic tree representation of this is illustrated in Figure 1-19.

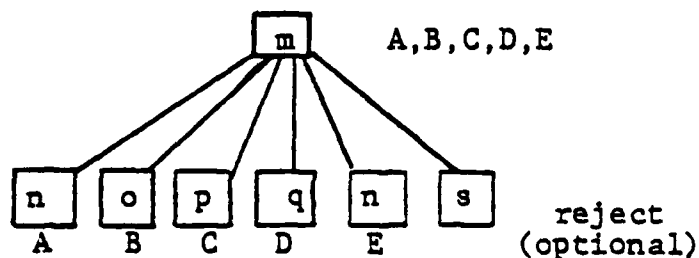


FIGURE 1-19
Complete Within-Group Logic Tree Node (5 classes)

This type of logic is more statistically based and requires less user interaction than group logic, where the analyst must determine the boundary(s) himself. MOOS does, however, allow modifications of these logics wherein the analyst may make a considerable number of subjective decisions. The three variations of complete within-group logic are Nearest Mean Vector (NMV) logic, pairwise logic, and closed decision boundary logic.

1.3.3.2.1 NMV Logic

Generalized NMV logic is a k-class classification technique; it classifies an unknown vector from the feature space according to a metric, which is computed from the unknown vector to the mean vectors of the k classes of the design set. The decision is in favor of the class which produces the minimum value of the metric. The generalized metric is:

$$d_i = \sqrt{(\underline{X} - \underline{M}_i)^T C_i^{-1} (\underline{X} - \underline{M}_i)}$$

where

\underline{X} = the l -dimensional unknown feature vector

\underline{M}_i = the l -dimensional mean vector for class i

C_i = an $l \times l$ matrix

If C_i is the covariance matrix for class i of the design set, then the metric is known as the Mahalanobis distance.

If C_i is the identity matrix, the metric is simply the Euclidean distance from the unknown vector to the mean vector of each class.

In MOOS, three basic options of NMV Logic are available; a reject strategy can be specified under each option. To reduce unnecessary calculation we use the square of the metric.

In the first option (simple NMV) C is the identity matrix, and the metric is computed in the form:

$$d_i = \sum_{j=1}^l (x_j - \mu_j^i)^2$$

where \underline{x} = unknown vector = $(x_1, x_2, \dots, x_\ell)$

\underline{M}_i = mean vector of class $i = (\mu_1^i, \mu_2^i, \dots, \mu_\ell^i)$

In the second option (weighting vectors), C_i is a diagonal matrix whose elements are the variances of the ℓ components of the design set samples of class i . The computational form of the metric used in this option is:

$$d_i = \sum_{j=1}^{\ell} \frac{(x_j - \mu_j^i)^2}{v_j^i}$$

where v_j^i = variance of j^{th} component of the i^{th} class

In the third option (weighting matrix), C_i is the covariance matrix of the design set samples of class i . The computation in this case involves the actual vector times matrix times vector multiplication, as defined by the generalized metric formula.

The optional reject strategy allows the user to specify a reject distance. In this case the decision strategy is: decide on the class j for which d_j is the minimum of all d_i , $i = 1, \dots, k$, if and only if d_j is less than the reject distance, otherwise reject. The user can specify a separate reject distance for each class, or he may use the same reject distance for all classes. This strategy may be used with any of the three metrics.

1.3.3.2.2 Pairwise Logic

In MOOS, pairwise logic is created by the routine fisher. This routine creates a one-space logic based on the Fisher direction (see section 1.3.2.2) for each possible pair of classes from among the classes present at the node for which the logic is being designed. Given N classes at the node, this will produce $N(N-1)/2$ class pairs. Each of these class-pair logics classifies (or can be thought of as producing a vote for) a vector as one or the other of these classes (or reject, depending upon the number of thresholds selected - see section 1.2).

In the evaluation of pairwise logic, the unlabeled vector is evaluated by each of the $N(N-1)/2$ class-pair logics and a "vote count" is kept for each class. After all class pairs have been evaluated, the vector is classified according to the vote counts. It is classified as belonging to that class which received the maximum vote count, provided this maximum is greater than or equal to a user-specified vote count threshold. In case of a tie for the maximum vote count, an attempt is made to break the tie by referring to the a priori class probabilities; if these are also equal, the vector is rejected.

The flow of pairwise logic evaluation is illustrated in Figure 1-21.

MOOS also allows the user, through the routine pair-mod, the capability of modifying each of the class-pair logics. The allowable types of logic are:

- 1) Fisher (1 to 5 thresholds)
- 2) Any arbitrary one-space projection vector
- 3) Optimal discriminant plane
- 4) Any arbitrary two-space plane
- 5) Boolean

1.3.3.2.3 Closed Decision Boundary Logic

A closed decision boundary logic strategy is implemented in the routine closedcn. This program creates an L-dimensional hyperregion to enclose each of the classes in the selected data set. Three types of hyperregion are available: hyperrectangular, hyperspherical, and hyperellipsoid.

The evaluation of hyperrectangular logic is performed as follows:

- 1) Project the unknown sample vector on the basis vectors.

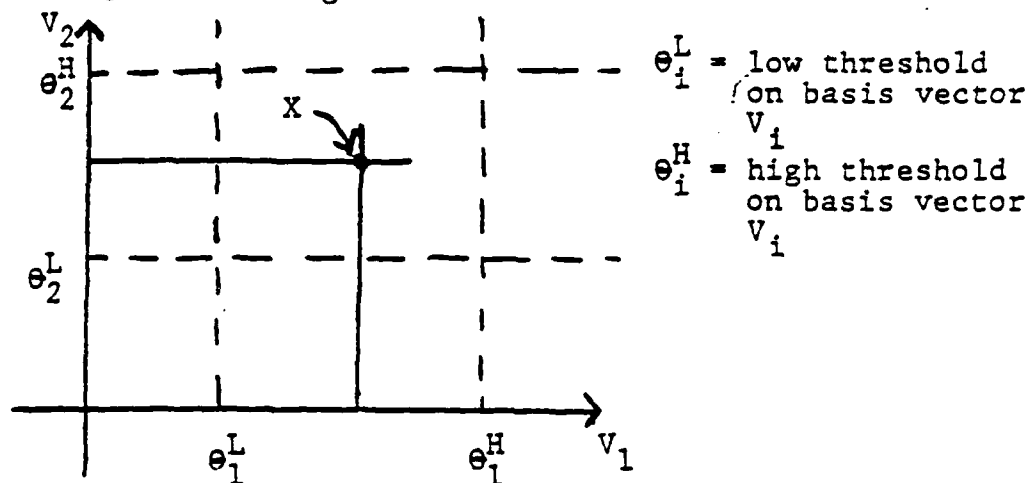
$$Y_j = \underline{X} \cdot \underline{V}_j$$

\underline{X} = the unknown vector

Y_j = the j th component of the projected vector

\underline{V}_j = the j th basis vector

- 2) The unknown vector is tested against a high and a low threshold along each basis vector. The vector is in the hyperrectangle if and only if its projection on each of the basis vectors is within the high and low thresholds for each basis vector. A two-dimensional case is illustrated in Figure 1-20.



The point X lies between both pairs of thresholds on the basis vectors and is therefore inside the hyperrectangle.

Figure 1-20

The evaluation of hyperspherical closed decision boundary logic is performed by calculating the Euclidean distance between the unknown vector and the center vector of the hypersphere. If this distance is less than or equal to the radius of the hypersphere, then the unknown vector is inside the hypersphere.

$$d^2 = \sum_{i=1}^L (X_i - M_i)^2$$

M_i = i^{th} component of the center vector of the hypersphere.

X_i = i^{th} component of the unknown vector.

d = Euclidean distance between X and M (d^2 is used rather than d to reduce computation).

The evaluation of hyperellipsoid closed decision boundary logic is performed by doing the following calculation for an unknown vector.

$$(\underline{X} - \underline{M})^T W (\underline{X} - \underline{M}) \leq C$$

\underline{X} = the unknown vector

\underline{M} = the center of the hyperellipsoid

W = an L-by-L weighting matrix

C = a size parameter analogous to the radius of a hypersphere.

If the above condition is met, the unknown vector lies inside the hyperellipsoid.

The weighting matrix W is determined as follows:

$$W = [B^T A B]^{-1}$$

B = an L-by-L matrix whose rows are the axis vectors of the hyperellipsoid (the only axis vectors currently implemented are the eigenvectors of the covariance matrix of the class)

A = an L-by-L diagonal matrix. A_{ii} = the length of the i^{th} axis of the hyperellipsoid

In the case where A is a diagonal matrix of eigenvalues and the center vector \underline{M} is the mean of a class: W is the inverse covariance matrix and C is the Mahalanobis distance.

Each class of a given data set may have any one of the three types of hyperregion surrounding it. Three cases arise depending on how many hyperregions an unknown vector falls into. If an unknown vector does not lie in any hyperregion, it is rejected. If an unknown vector falls in one hyperregion, it is assigned to the class associated with that hyperregion. If an unknown vector lies in more than one hyperregion (referred to as overlap in further discussion), it is rejected unless the user has specified otherwise.

"Overlap" vectors may be placed in a new data tree, and further classification logic developed on the new data tree to reduce the number of rejections produced by closed decision boundary logic. This is a non-standard approach in that a data set must be passed against two independent logic trees to produce the final classification results.

When an unknown vector is rejected due to an overlap condition, some useful information may be retained. If the vector fell into only a few of the possible hyperregions, at least the number of choices as to which class the vector really belongs has been narrowed. This partial classification information may be utilized by Fisher pairwise logic.

The evaluation of Fisher pairwise logic performed on a data tree consisting of "overlap" vectors differs from the usual pairwise evaluation. Each unknown vector is tested only by pairwise decisions involving the possible classes indicated by closed decision boundary logic. Partial classification information obtained from a closed decision boundary evaluation may be utilized only by pairwise logic.

1.3.3.3 Logic Evaluation Outputs

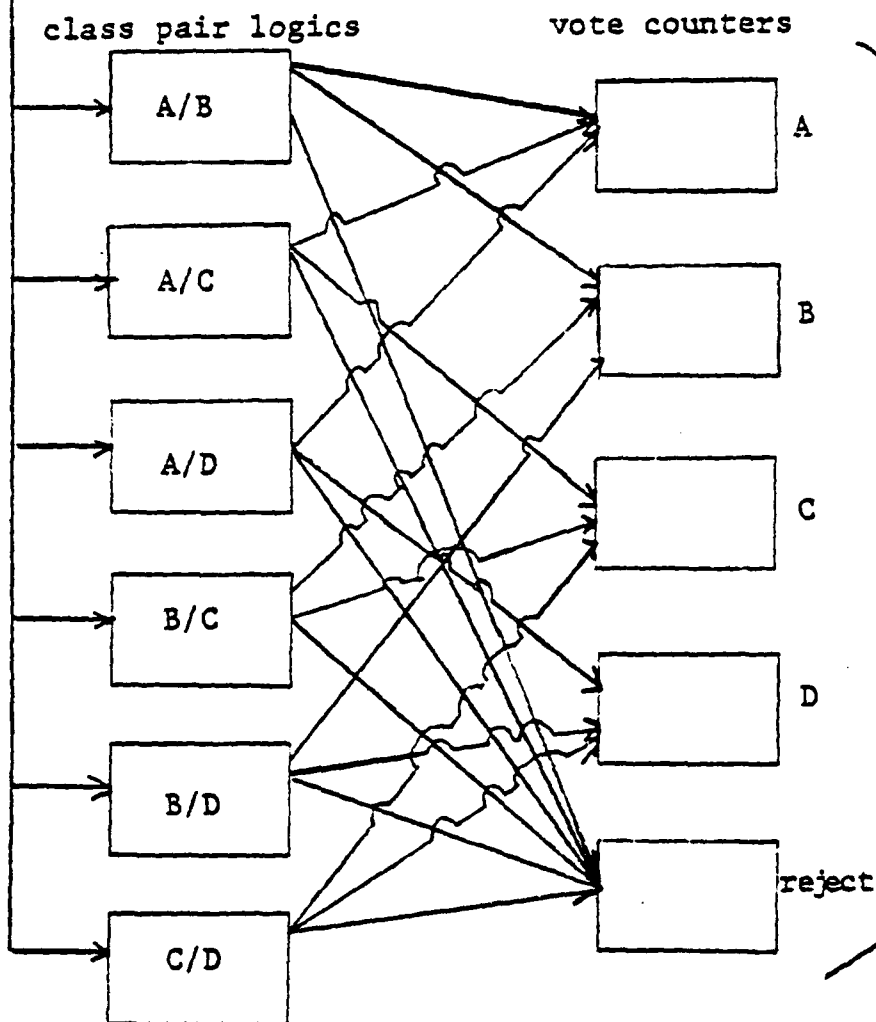
This section describes the various types of confusion matrix displays produced by MOOS, and gives some general guidelines for interpreting these specialized formats.

1.3.3.3.1 Confusion Matrix for Temporary Between-Group Logic.

The following confusion matrix format is produced by any one-space group logic, two-space group logic, or Boolean partition logic.

Referring to Figure 1-22, the first few lines of output represent the user interaction with the logic design routine (in this case a two-space group logic). The heart of the display consists of a matrix format in which the columns are associated with nodes in the logic tree structure, and the rows correspond to the data classes in the data set being evaluated. Any particular element of the matrix is the number of vectors from a given class which were assigned to a particular logic node. The leftmost column of numbers always refers to the node on which logic was designed. To the right of and below the matrix are various totals and percentages designed to aid the analyst in the interpretation of these results.

unlabeled vector



classify
according
to class
with max.
vote count
iff max.
 $\geq T$, other
wise reject

The output of each class-pair logic can be one of the following:

- a vote for no class (reject)
- or
- a vote for the first class of the pair
- or
- a vote for the second class of the pair

FIGURE 1-21
PAIRWISE LOGIC (4 CLASSES)

Enter the display symbols of the classes on the convex side of boundary 1
 (on one line - no delimiters)
 coswr
 the logic node assigned to these dataclasses is 2
 Enter the display symbols of the classes on the excess side of the boundary(a)
 avco
 the logic node assigned to these classes is 3

| | logic node | | | | | | |
|------|------------|-------|-------|-----|-------|-----|------|
| | 1 | 2 | 3 | cor | xcor | err | xerr |
| soys | 81 | 61 | 0 | 81 | 100.0 | 0 | 0.0 |
| corc | 60 | 23 | 37 | 60 | 100.0 | 0 | 0.0 |
| oato | 65 | 53 | 12 | 65 | 100.0 | 0 | 0.0 |
| whaw | 60 | 60 | 0 | 60 | 100.0 | 0 | 0.0 |
| clou | 62 | 0 | 62 | 62 | 100.0 | 0 | 0.0 |
| alfa | 54 | 0 | 54 | 54 | 100.0 | 0 | 0.0 |
| ryer | 60 | 60 | 0 | 60 | 100.0 | 0 | 0.0 |
| tot | 422 | 257 | 165 | 422 | 100.0 | 0 | 0.0 |
| cor | 422 | 257 | 165 | | | | |
| xcor | 100.0 | 100.0 | 100.0 | | | | |
| err | 0 | 0 | 0 | | | | |
| xerr | 0.0 | 0.0 | 0.0 | | | | |

Do you want a hardcopy with a listing of misclassified vectors?

Figure 22: Confusion Matrix for Temporary Between-Group Logic

If the user decides to produce a high-speed printer copy of this confusion matrix, he may choose to get a listing of all incorrectly classified vectors. Each error is listed by data class, vector identification number, and the logic node to which it was assigned.

1.3.3.3.2 Confusion Matrix for Temporary Within-Group Logic

The following confusion matrix format is produced by the within-group logic routines nmv, fisher, and closedcn. Referring to Figure 1-23, the first two lines of output describe the type of within-group logic being evaluated, the data set on which logic was designed, and the number of dimensions. The heart of the display consists of a matrix format in which the column labels correspond to the data classes of the data set being evaluated, and the row labels are associated with the classes in the data set on which logic was designed. Any particular element of this matrix is the number of vectors from a given data class which were assigned to a particular logic node. (In the case where all classification was correct, all off-diagonal elements of the matrix would be zero.) Below this matrix are various totals and percentages designed to aid the analyst in the interpretation of his results.

If the user decides to produce a high-speed printer copy of a confusion matrix, he may choose to get a listing of all incorrectly classified vectors. For both nmv, fisher, and closedcn, each error is listed by data class, vector identification number and the logic node to which it was assigned. The following additional useful information is listed for each vector:

In the case of nmv, for each misclassified vector, the distance to the true class and the distance to the assigned class is listed. If the vector was rejected, the distance to the closest class is listed rather than the distance to the assigned class.

In the case of fisher, the first additional line usually begins with the phrase "lost to:" followed by a list of display symbols. Each display symbol refers to an incorrect pairwise decision involving the true class. If the vector was assigned to the wrong class by a pairwise decision box, the display symbol of the incorrect class is listed. An "r" in parentheses immediately following a class symbol indicates that the vector was rejected, not misclassified, by that decision box. The second line contains a list of vote counts for the given vector, in order of ascending logic node number. The last vote count listed is always the value of the reject vote count. If there was a tie situation, the first additional line of output is preceded by "tie" or "favorably broken tie" (see Section 1.3.3.2.2).

Closed decision boundary logic (closedcn) lists the type of hyperregion associated with the true class. If a vector falls into more than one hyperregion, the names of the classes associated with those hyperregions are also listed.

Partial Polytube Evaluation: nearest sum logic node 1
 Number of dimensions = 6

pairmod
 sumrycm
 displacm
 hrdepycm

| | | true class | | | | | |
|-------|------|------------|------|------|------|------|------|
| | | soya | corr | onto | whw | clou | alfa |
| soya | 65 | 3 | 1 | 0 | 0 | 0 | 0 |
| corr | 55 | 55 | 0 | 0 | 0 | 0 | 0 |
| onto | 0 | 0 | 59 | 0 | 0 | 0 | 1 |
| whw | 0 | 0 | 2 | 60 | 0 | 0 | 0 |
| clou | 0 | 0 | 0 | 0 | 56 | 3 | 0 |
| alfa | 0 | 0 | 0 | 0 | 2 | 46 | 0 |
| ryer | 0 | 0 | 1 | 0 | 0 | 0 | 52 |
| rejt | 4 | 2 | 2 | 0 | 4 | 5 | 1 |
| totl | 61 | 60 | 65 | 60 | 62 | 54 | 60 |
| corr | 55 | 55 | 59 | 60 | 56 | 46 | 52 |
| xcor | 90.2 | 91.7 | 90.8 | 90.0 | 90.3 | 85.2 | 96.7 |
| error | 2 | 3 | 4 | 0 | 2 | 3 | 1 |
| xerr | 3.3 | 5.0 | 6.2 | 0.0 | 3.2 | 5.6 | 1.7 |
| rejt | 4 | 2 | 2 | 0 | 4 | 5 | 1 |
| xrejt | 6.6 | 3.3 | 3.1 | 0.0 | 6.5 | 9.3 | 1.7 |

total number of vectors = 428
 overall correct 389 for 92.18%
 overall error 15 for 3.55%
 overall reject 18 for 4.27%

Current
 Option:
 fisher
 Dataset:
 nearest
 test

Figure 1-23: Confusion Matrix for Temporary
 Within-Group Logic

1.3.3.3.3 Confusion Matrix for Overall Logic Evaluation

The confusion matrix produced by overall logic evaluation (logicevl) is identical in format to the matrix produced by the temporary within-group logic routines nmv, fisher, and closedcn. If the user chooses to list this matrix on the high-speed printer, a list of all incorrectly classified vectors may be produced in the format described previously.

1.3.3.3.3.1 Reassociated Names

Additional flexibility is made possible for the overall logic evaluation of an independent data set by the use of the reassociated names capability. Utilizing the routine logicevl, any data set may be tested against logic designed on any other data set of equal dimensionality. However, the totals and percentages correct listed below the matrix will be useful only if the names of the data classes on which logic was designed are the same as the names of the data classes being evaluated. This may be accomplished through the use of reasname, which allows the user to tag logic nodes with any reassociated names.

In a case where two or more logic nodes have been given the same reassociated name, the totals below the matrix are formed by adding the confusion matrix entries for these logic nodes.

If reassociated names have been added to the logic tree, logicevl asks the user whether the reassociated names are to be used in the confusion matrix printout. If the response is yes, the reassociated names will be used in place of the original design names; if more than one logic node has the same reassociated name, only one entry will appear in the confusion matrix for that name. In all cases where reassociated names have been added to a logic tree, they will be used to determine whether the vectors in the data set being evaluated have been assigned correctly.

One further embellishment has been added to the reassociated name capability. If sense switch 2 is set prior to overall logic evaluation, the test of correctness is simply made on the display symbols of the classes involved, rather than on the entire four-character names.

1.3.4 Boolean Partitions

MOOS has also implemented a user capability for Boolean defined partitions of the feature space. This capability can be used in structure analysis, group logic and pairwise logic.

This is implemented through utilization of the PL/1 compiler under MULTICS. As a result of the flexibility of MULTICS, the analyst can write any Boolean statement (one that can be evaluated as true or false), provided that it is a legal PL/1 statement and that it conforms to certain conventions for referencing feature vector components, and then use that statement as the basis for a transformation or a partition.

1.3.5 Measurement Transformations

In addition to a measurement reduction transformation (trnsform) performed in conjunction with measurement evaluation computations, a data set within MOOS may be transformed by any of the following three independent transformations; normxfm, eigentrn, or measxfm. Upon execution of any of these algorithms, every vector in the selected data set is transformed and a new tree is created from the transformed vectors. The new tree will have the same structure as the original tree.

1.3.5.1 The Normalization Transformation

The normalization transformation, normxfm, determines the standard deviation along each coordinate measurement of the selected data set. Each vector component within the data set is then modified by dividing it by its corresponding standard deviation. The resulting normalized data set will have unit variance along each coordinate measurement.

In some cases, normalization may be necessary to ensure that the various numerical calculations performed by MOOS (e.g. matrix inversions) are sufficiently accurate.

1.3.5.2 The Eigenvector Transformation

The eigenvector transformation, eigentrn, computes the eigenvectors of the covariance matrix of the selected data set (see Section 1.3.2). The user is then given the option of mapping the selected L-dimensional data set onto an M-dimensional eigenvector subspace ($M \leq L$) by selecting the M eigenvectors corresponding to the M largest eigenvalues. The resulting M-dimensional subspace provides a least squares fit to the selected data set, since the sum of the squared residual

distances from the subspace is minimized. The error in fitting the data is given by summing the remaining eigenvalues:

$$\text{Squared Fitting Error} = \sum_{j=M+1}^L \lambda_j$$

The transformation essentially involves an orthonormal rotation of the basis vectors of the data set until they are aligned with the eigenvectors.

This technique has proven useful both as a research tool and as an aid to structure analysis and logic design. Measurement reduction may also be performed through use of the eigenvector transformation.

1.3.5.3 The Measurement Compiler Transformation

By using the routine measxfrm, the MOOS user may define new features which are functions of the original L measurements. The capability of the MULTICS PL/1 compiler is utilized in that any statements allowed by PL/1 may be used for this transformation.

The measurement compiler option provides the MOOS user with a practically unlimited capability for defining both linear and nonlinear transformations. Once the new features have been defined, the system will execute the transformation, thereby generating a new data tree whose vectors have the new user-defined features as their components.

1.3.5.4 Measurement Reduction Transformations

The MOOS system provides three methods for selecting a projection of the "current data" onto a coordinate subspace in conjunction with the three methods for evaluating the discriminatory value of each measurement (Section 1.3.1). Each of these measurement evaluation algorithms (dscrmeas, probconf, features) produces rank order displays of the L measurements according to a user-specified criterion. The user may select specified measurements from the data set via the commands sel\$ and un\$. The measurements which are chosen for retention define the coordinate subspace and the desired linear transformation. The user then calls the measurement reduction transformation routine transform to implement the specified transformation, thereby creating a tree identical in structure, but containing vectors of fewer measurements than the original data tree.

REFERENCES

1. Kanal, Laveen N., "Interactive Pattern Analysis and Classification Systems: A Survey and Commentary", IEEE Proceedings, Vol 60, No. 10, pp. 1200-1215, October 1972.
2. Sammon, J.W. Jr., "Interactive Pattern Analysis and Classification", IEEE Transactions on Computers, Vol C-19, pp. 594-616, July 1970.
3. Simmons, E.J. Jr., "Interactive Pattern Recognition - A Designers Tool", AFIPS Conference Proceedings, Vol 42, pp. 479-483, June 1973.
4. Marill, T. and Green, D.M. "On the Effectiveness of Receptors in Recognition Systems," IEEE Transactions on Information Theory, Vol. IT-9, pp. 11-17, January 1963.
5. Kadota, T.T. and Shepp, L.A. "On the Best Finite Set of Linear Observables for Discriminating Two Gaussian Signals," IEEE Transactions on Information Theory, Vol. IT-13, pp. 278-284, April 1967.
6. Sammon, J.W. Jr., "An Optimal Discriminant Plane", IEEE Transactions on Computers, Vol. C-19, pp. 826-829, September 1970.

APPENDIX B

OLPARS USER NOTES

B.1 DATA PARTITIONS (BOUNDARIES)

DRAWBNDY lets an OLPARS user partition a one-space or two-space projection for the purpose of restructuring a data class or designing group logic. The partitions are used to define a region in which a vector is located.

For a one-space projection, a partition is created by designating a threshold value along the baseline (x - axis) of the display. The line perpendicular to the baseline at the threshold value represents the partition (boundary).

For a two-space projection, a partition is created by drawing a convex boundary containing up to five line segments. A boundary is convex if a convex polygon can be created when lengthening the end line segments to meet the display borders. (A convex polygon lies completely on one side of any side (line segment) extended.) The convex side of the boundary is designated by the analyst by entering a "convex point". The so-called "convex point" indicates the interior region or "convex region" of the polygon. (Note, any

boundary with less than three line segments is convex).

Figure B-1 shows, for one and two-space displays, the regions defined by creating one and two boundaries.

B.2 EVALUATION PROCEDURE

Assigning a data vector to a region

When a user draws boundaries on a display, (s)he has an idea about how a class or group of classes should be divided (partitioned). There is a visual perception of which vectors belong to each region. In order to assure that the vectors are assigned to the regions as expected, the user must be aware of the method (algorithm) used by the OLPARS programs for assigning a vector to a region. When restructuring a data class or designing group logic, this information is necessary so the user can make the appropriate associations between display regions and data classes. The following text describes the method in which vectors are assigned to display regions.

B.2.1 ONE-SPACE

One Boundary (One Threshold)

If the data vector is on the left side of the boundary, the vector is assigned to the LEFT REGION; otherwise, the vector is assigned to the RIGHT REGION.

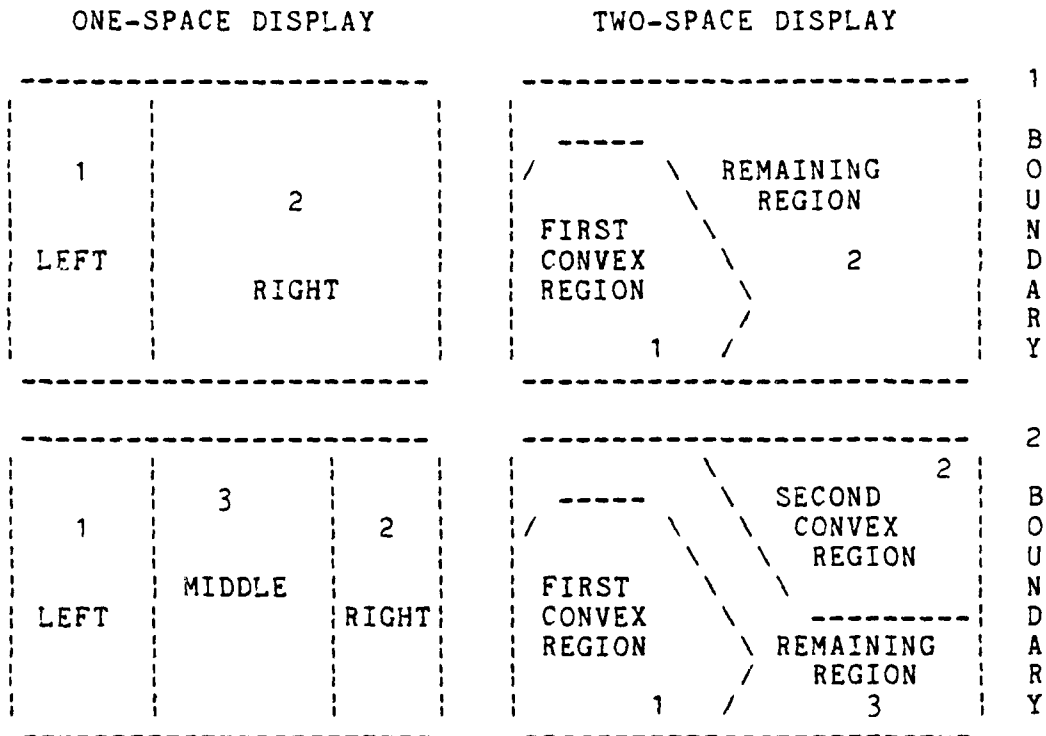


Figure B-1 User Defined Regions On
One and Two Space Displays

Two Boundaries (Two Thresholds)

The order in which the thresholds are entered by the user is not important. The smaller (or left-most) threshold determines the left-most boundary, and the larger (or right-most) threshold determines the right-most boundary. If the data vector is on the left side of the left-most boundary, the vector is assigned to the LEFT REGION. Otherwise, the vector is compared against the right-most boundary. If the vector is on the left side of the right-most boundary, the vector is assigned to the MIDDLE REGION; otherwise, it is assigned to the RIGHT REGION. /

B.2.2 TWO-SPACE

For two-space displays, the OLPARS programs which assign a vector to a region assume (1) that the boundaries are convex, and (2) that the so-called "convex point" is on the convex side of the boundary with which it is associated. If these assumptions are not met, the vectors may not be assigned to regions as expected by the user.

One Boundary

If the data vector is on the convex side of the boundary, the vector is assigned to the FIRST CONVEX REGION; otherwise, it is assigned to the REMAINING REGION (sometimes called EXCESS REGION).

Two Boundaries

The order in which the boundaries are entered by the user is important in the evaluation process. The first boundary entered will be called "boundary 1," and the second boundary entered will be called "boundary 2." If the data vector is on the convex side of boundary 1, the vector is assigned to the FIRST CONVEX REGION. Otherwise, the vector is compared against boundary 2. If the vector is on the convex side of boundary 2, the vector is assigned to the SECOND CONVEX REGION. Vectors outside of both convex regions are assigned to the REMAINING REGION.

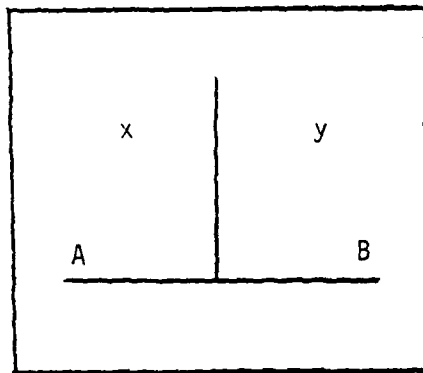
The most important fact for the user to be aware of is that the evaluation begins with boundary 1. That is, vectors are assigned to the first convex region before they are assigned to the second convex region, and assigned to the second convex region before they are assigned to the remaining region. Therefore, the order in which the boundaries are entered may affect the vector assignment, especially when two boundaries cause one region to be contained within another region, or when two boundaries intersect within the region of vectors being separated. The analyst must understand the evaluation process so that the resulting vector-to-region assignments can be predictable. (See the examples which follow.)

OLPARS User Notes -- B
EXAMPLES

B.2.3 EXAMPLES

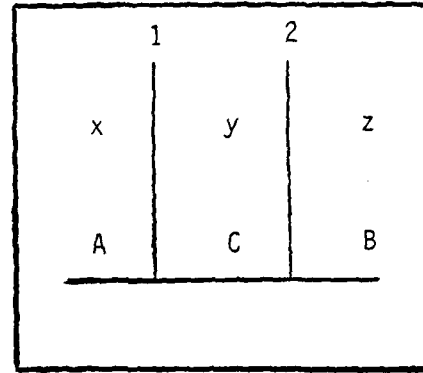
Figure B-2 shows examples of vector-to-region assignments. The individual displays are labelled as follows: Region 'A' is the LEFT or FIRST CONVEX REGION; Region 'B' is the RIGHT or SECOND CONVEX REGION; Region 'C' is the MIDDLE or REMAINING REGION; The numbers on the display diagrams are boundary numbers; and t, u, w, x, y, and z are vectors.

1 - Space 1 Boundary



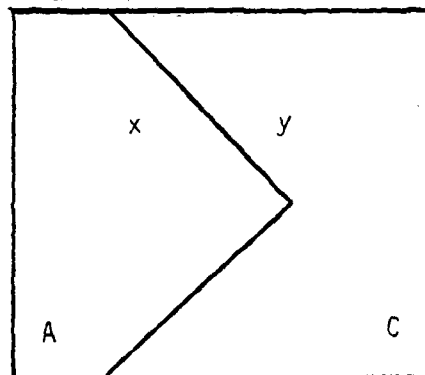
vector x to region A
vector y to region B

1 - Space 2 Boundaries



vector x to region A
vector y to region C
vector z to region B

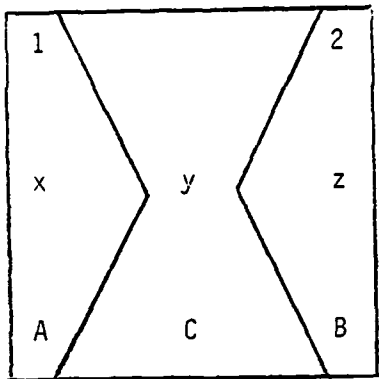
2 - Space One Boundary



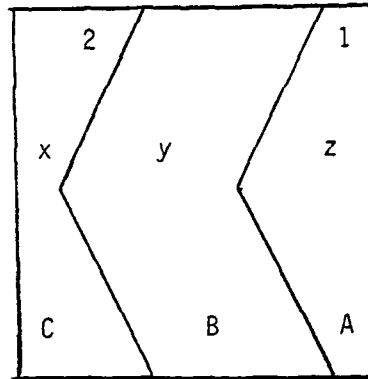
vector x to region A
vector y to region C

FIGURE B-2

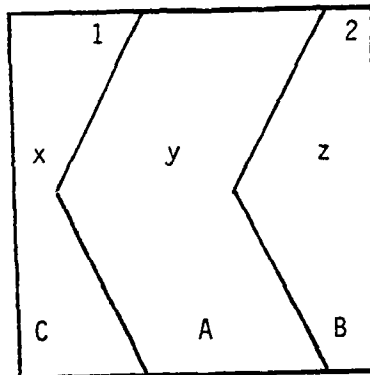
2 - Space 2 Boundaries



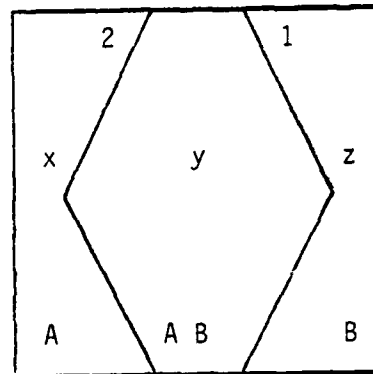
vector x to region A
vector y to region C
vector z to region B



vector x to region C
vector y to region B
vector z to region A



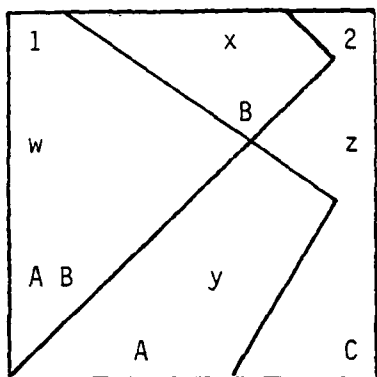
vector x to region C
vector y to region A
vector z to region A



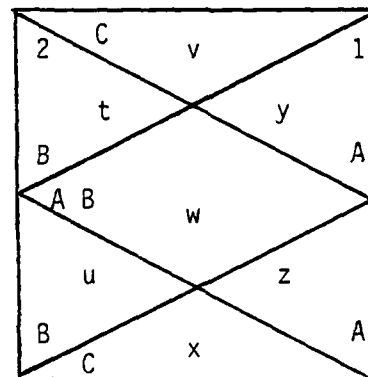
vector x to region A
vector y to region A
vector z to region B

FIGURE B-2 (continued)

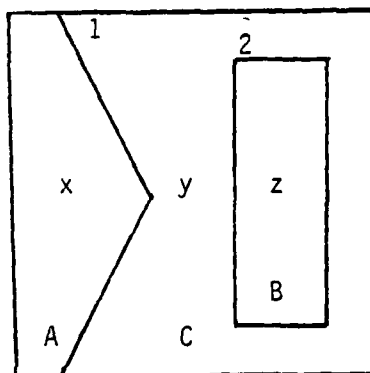
2 - Space 2 Boundaries



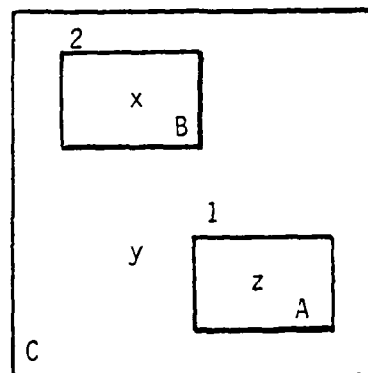
vector w to region A
vector x to region B
vector y to region A
vector z to region C



vector t to region B
vector u to region B
vector v to region C
vector w to region A
vector x to region C
vector y to region A
vector z to region A



vector x to region A
vector y to region C
vector z to region B



vector x to region B
vector y to region C
vector z to region A

FIGURE B-2 (continued)

INDEX

| | |
|--|---------------------------|
| ANYTHING (Display available OLPARS commands) | 119 |
| APPEND (Add node from one tree to another) | 120 |
| Assigned classes | 28, 315 |
| Associated class names | 315 |
| Between-group logic | 81, 85, 87, 105, 109, 315 |
| BINWIDTH (Alter bin size on 1-space display) | 122 |
| BYEOLP (OLPARS "logout" function) | 126 |
| CDEFAULT (Change or Set CM File Default Values) | 127 |
| CDISPLAY (Change One- or Two-Space Display) | 131 |
| Class display symbol | 34, 315 |
| Class pair | 34, 61, 66, 70, 315 |
| Class select list | 10, 316 |
| Classification table | 316 |
| Cluster plot | 19, 316 |
| COMNOD (Combine two or more lowest data tree nodes) | 132 |
| Complete within-group logic | 316 |
| Completed logic node | 46, 316 |
| Completed logic tree | 91, 97, 316 |
| Confusion matrix | 11, 85, 97, 100, 316 |
| Convex point | 316, B-2 |
| Convex region | 316, B-2 |
| Convex side | 316, B-2 |
| Coordinate projection | 7, 53, 77, 316 |
| Covariance matrix | 9, 54, 104, 316 |
| CRANDTS (Create Random Data Test Set) | 135 |
| CREATLOG (Create logic for 1,2 space projections) | 137 |
| CSCALE (Change scaling for 1,2-space displays) | 139 |
| Current data set | 5, 316 |
| Current logic | 5, 317 |
| Current option | 3, 317 |
| Data class | 3, 317 |
| Data partition | 317 |
| Data set dimensionality | 317 |
| Data tree | 11 |
| Data vector | 4, 317 |
| DBNDY (Delete all existing display boundaries) | 140 |
| DDATANOD (Delete a lowest node from a data tree) | 141 |
| DDATATREE (Delete data tree from user directory) | 144 |
| DDSUBSTR (Delete data tree substructure) | 145 |
| Design data set | 10, 51, 317 |
| Discriminant measure | 61 to 62, 65, 317 |
| DLOGTREE (Delete logic tree from user directory) | 148 |
| DLSUBSTR (Delete logic tree substructure) | 149 |
| DRAWBNDY (Partition a data projection) | 152 |
| DRAWLOG (Display structure of a logic tree) | 156 |
| DRAWTREE (Display structure of a data tree) | 159 |
| DSCRMEAS (Compute measurement evaluation statistics) | 161 |

OLPARS User Manual
INDEX

| | |
|---|---------------------------------|
| DTRENAME (Rename an OLPARS data tree) | 164 |
| DVEC (Delete vectors from data tree class) | 165 |
| Eigenvector projection | 54, 72, 77, 104, 317 |
| EIGNXFRM (eigenvector data transformation) | 168 |
| Excess measurement mode | 6, 317 |
| Feature (measurement) | 317 |
| Feature extraction | 48, 103, 317 |
| FILEIN (Create OLPARS data tree from a text file) | 171 |
| FILEOUT (Place OLPARS data tree vectors into text file) | 175 |
| FISHER (Create Fisher logic) | 177 |
| Fisher direction (projection) | 318 |
| FISHMOD (Modify vote or threshold count of Fisher logic) | 183 |
| Global scale | 318 |
| HELP (Provide help on an OLPARS subject or command) | 185 |
| Ignored measurements | 8 |
| Incomplete logic | 43, 318 |
| Incomplete logic node | 43, 318 |
| INTENSIFY (Highlight class(es) currently displayed) | 187 |
| L1ASDG (1-Space Assigned Discriminant Groups for L.D.) | 189 |
| L1CRDV (1-Space Coordinate Projection for Logic Design) | 193 |
| L1EIGV (1-Space Eigenvector Projection for Logic Design) | 195 |
| L2ASDG (2-Space Assigned Discriminant Groups for L.D.) | 199 |
| L2CRDV (2-Space Coordinate Projection for Logic Design) | 203 |
| L2EIGV (2-Space Eigenvector Projection for Logic Design) | 205 |
| L2FSHP (2-Space Fisher Direction Projection for L.D.) | 209 |
| LISTLOGS (Display user logic tree names) | 212 |
| LISTTREES (Display user data tree names) | 213 |
| LOGEVAL (Evaluate current logic using current data set) | 214 |
| Logic | 5, 318 |
| Logic design | 7, 10, 81, 318 |
| Logic tree | 11 |
| LTRENAME (Rename an OLPARS logic tree) | 219 |
| Macro plot | 14, 53, 318 |
| MAKETREE (Create data tree from existing trees) | 220 |
| MATRIX (Save or delete matrices in the SM file) | 223 |
| MATXFRM (Perform matrix trans. to create a new data tree) | 230 |
| Mean vector | 318 |
| Measurement evaluation | 7, 12, 61, 70, 72, 75, 104, 318 |
| Measurement transformation | 76, 80, 105, 318 |
| MEASXFRM (User specified data measurement transformation) | 232 |
| Micro plot | 14, 87, 318 |
| MOVEC (Move vector in 2-space coordinate vector projection) | 236 |
| Multimodal | 53 to 54, 57, 81, 104, 318 |
| NAMELOG (Create a new logic tree) | 238 |
| Nearest mean vector | 28, 90, 318 |

| | |
|--|-----------------------------|
| NMEVAL (Evaluate nearest mean vector logic node) | 239 |
| NMV (Create nearest mean vector logic) | 240 |
| NMVMOD (Modify nearest mean vector logic) | 243 |
| NNMOD (Modify nearest neighbor logic) | 245 |
| NORMXFRM (Create normalized version of current data set) | 248 |
| NRSTNBR (Create nearest neighbor logic) | 250 |
| One-space | 11, 14, 53, 87, 319 |
| OPTIMLMOD (Create optimal discrim. logic at Fisher node) | 253 |
| Option list | 6 |
| Option list (menu) | 319 |
| Overall logic evaluation | 12, 91, 97, 319 |
| Overlap graph | 50, 53, 104, 319 |
| Pairwise logic | 90, 319 |
| Partial logic evaluation | 319 |
| Projection | 319 |
| PROJMN (Display mean vector of projected classes) | 257 |
| Prompt mode | 9, 319 |
| PRTCM (Print confusion matrix) | 258 |
| PRTDS (Print data set) | 259 |
| PRTIDX (Print vector ids. and vector counts) | 263 |
| PRTLOG (Print information for any logic tree) | 266 |
| PWEVAL (Evaluate Fisher pairwise logic node) | 269 |
| RANK (Display user-specified type of ranking) | 270 |
| Rank order | 11 |
| Rank order display | 12, 32, 35, 61, 66, 70, 320 |
| Raw measurement | 50, 320 |
| RDISPLAY (Redisplay 1,2-space proj., confusion matrix) | 274 |
| REASNAME (Change reasssociated names in current logic) | 275 |
| Reassociated class name | 98, 320 |
| Rectangular scale | 22, 320 |
| REDRAW (Put up an existing boundary or threshold) | 278 |
| REPROJECT (Reproject data on different eigenvectors) | 279 |
| RESTRUCT (Restructure one class of a data tree) | 281 |
| Restructure | 57, 104, 320 |
| S1CRDV (1-Space Coordinate Projection) | 284 |
| S1EIGV (1-Space Eigenvector Projection) | 285 |
| S2CRDV (2-Space Coordinate Projection) | 288 |
| S2EIGV (2-Space Eigenvector Projection) | 289 |
| S2FSHP (2-Space Fisher Direction Projection) | 292 |
| Scale mode | 17, 22, 320 |
| Scale type | 17, 22, 320 |
| SCALRET (Return to original display scale) | 295 |
| SCALZM (Perform zooming on 1,2-space displays) | 296 |
| Scatter matrix | 320 |
| Scatter plot | 19, 66, 85, 320 |
| SELECT (Select class symbols to display) | 298 |
| Senior node | 3, 5, 37, 43, 320 |
| SETDS (Set Current Data Set Name) | 300 |
| SETLOG ('Restore' an existing logic) | 302 |

OLPARS User Manual
INDEX

| | |
|--|---------------------------|
| SLCTMEAS (Select meas. in a rank order display) . . . | 303 |
| Square scale | 22, 320 |
| Statistical validity of user data set | 10 |
| Structure analysis | 7, 12, 53, 104, 321 |
| SUMMCM (Display confusion matrix summary) | 306 |
| Test data set | 10, 51, 96, 106, 321 |
| THRESHMOD (Modify thresholds in Fisher logic node) . . | 307 |
| TRANSFRM (Create new data tree using selected meas.'s) | 310 |
| True classes | 28, 321 |
| Two-space | 11, 17, 19, 53 to 54, 321 |
| Unimodal | 53, 61, 81 to 82, 321 |
| UNION (Select meas. in a rank order display) | 312 |
| Union by class | 61, 66, 70, 321 |
| Union by class pair | 61, 66, 70, 321 |
| Vector identifier | 8, 321 |
| Within-group logic | 81, 90, 105, 109 |
| Zoom scale | 17, 22, 321 |

FILMED

9-8